

ENT SENIOR DESIGN PROJECT REPORT

Internet of Things Proof of Concept

Submitted to

Professor David Goodman
Engineering Technology Department
by
Travis Cucore

December 4, 2018

Issued By:	Approved By:	Effective Date:	Page 2 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

ABSTRACT

The EdgeNet platform was conceived of to satisfy graduation requirements and demonstrate a breadth and depth of knowledge capable of working with embedded systems and supporting their integration with larger software platforms. The project is justified as a valid senior design supported by several senior level CpET courses described in the introduction of this report.

The EdgeNet IoT platform provides a flexible and brandable platform for connecting consumers and business with devices they wish to monitor or manipulate remotely. Built using industry standard languages and frameworks, consumer facing applications share a common code-base, reducing costs associated with code maintenance and development.

The platform is architected around the MQTT (Message Queuing and Telemetry Transport) protocol version 3.1.1. However, the discovery of “Doze Mode,” a feature of Android API versions 23 and above, has complicated the implementation of wireless communication for consumer facing applications. To get around this, the project must be rearchitected to comply with application service windows.

Issued By: Travis Cucore	Approved By: Travis Cucore	Effective Date: 10/31/2018	Page 3 of 91
		Document No.: 0001	Version: 01

Internet of Things Proof of Concept

Issued By: Travis Cucore	Approved By: Travis Cucore	Effective Date: 10/31/2018	Page 4 of 91
		Document No.: 0001	Version: 01

Internet of Things Proof of Concept

REVISION HISTORY

Version	Date	Revised by	Description
1.0	4 October 2018	Travis Cucore	Initial Draft
2.0	31 October 2018	Travis Cucore	First Draft for Peer Review
3.0	31 October 2018	Travis Cucore	Updated to Latest Version of Word
4.0	01 November 2018	Travis Cucore	Added Theory of operation section
5.0	11 November 2018	Travis Cucore	Added Completed Theory of operation section and started Code Characterization section.
5.1	4 December 2018	Travis Cucore	Final Version

Issued By:	Approved By:	Effective Date:	Page 5 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

Table of Contents

ABSTRACT.....	2
REVISION HISTORY.....	4
1. INTRODUCTION	6
1.1 Problem Statement.....	7
1.2 System Overview	7
1.2.1 Specifications.....	7
1.2.2 Summary of Core Components	7
2. REFERENCED DOCUMENTS	9
3. SYSTEM-WIDE DESIGN DECISIONS	9
3.1 Hardware.....	9
3.2 Firmware	9
3.3 Android Application	9
4. SYSTEM ARCHETECTURE	10
4.1 Overview of MQTT Protocol	10
4.1.1 Connectivity.....	10
4.1.2 Packet Structure	12
4.2 Payload Structure	13
4.3 Android Application	13
4.3.1 Libraries in Use	14
4.3.2 Classes.....	14
4.3.3 Views	16
4.3.4 Localized Interfaces	22
4.3.5 Data Bindings.....	23
4.3.6 Doze Mode	24
4.4 Firmware	25
4.4.1 Atmel Software Framework.....	25
4.4.2 WiFi Stack	26
4.4.3 Abstractions.....	26
4.4.4 PWM Driver	26
4.4.5 MQTT Client.....	28
5. CODE CHARACTERIZATION	33
5.1 Firmware	33
5.1.1 MCQTT.h.....	33
5.1.2 MCQTT.c	38

Issued By:	Approved By:	Effective Date:	Page 6 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

5.1.3 PWM.h	47
5.1.4 PWM.c	47
5.1.5 main.h	49
5.1.6 main21.c	51
5.2 Android Application	57
5.2.1 Views	57
5.2.2 Database Wrappers	79
5.2.3 MQTT Wrapper	81
5.2.4 Data Objects	87
5.2.5 Local Resource Interface	88
5.2.6 App.cs	89
5. CONCLUSIONS AND RECOMMENDATIONS	91
6. GLOSSARY OF TERMS	Error! Bookmark not defined.
References	91

1. INTRODUCTION

As a CpET (Computer Engineering Technology) student, I have focused on acquiring the skills required to develop and integrate with embedded systems. The following courses are provided as justification for this project.

ECET 309 – Advanced Microcontrollers

Advanced Microcontrollers provided advanced embedded C-language programming techniques and practical experience understanding how to approach a microcontroller of any architecture to deliver firmware. This project deploys an embedded MQTT client and interfaces with the standard Atmel Software Framework (ASF) demonstrating an understanding of firmware/embedded software stack. Further, a PWM (Pulse Width Modulation) driver was written and interfaced with the embedded MQTT client to provide an observable output demonstrating an understanding of low-level programming and how to implement and interface with abstraction.

ECET 334 – Advanced Object-Oriented Programming

Advanced Object-Oriented Programming covered the object-oriented programming (OOP) for embedded systems. Communication systems and structures were discussed and applied to embedded platforms. The Xamarin Forms development framework was heavily discussed in the context of cross-platform development to discover the utility in developing portable code. This project deploys an Android application developed on the Xamarin Forms framework using .Net

Issued By:	Approved By:	Effective Date:	Page 7 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

core demonstrating an understanding of OOP concepts and cross-platform development ideologies.

ECET 499 – Free Study (Data-Structures)

Free Study allowed the exploration of concepts required to have data persist between application starts. This project demonstrates persistent data and data-binding throughout the mobile application deploying a sqlite library and a localized interface to gain access to device specific hardware and create, modify, and delete databases on those local resources.

1.1 Problem Statement

The Internet of Things (IoT) landscape is an increasingly complex web of technology integrations. This project serves the purpose of demonstrating a marketable skillset for finding employment somewhere along the IoT software stack. To that end, core competencies required for the development of Internet-connected, customer-facing, and sometime disjointed systems were used to develop an end-to-end IoT proof of concept.

1.2 System Overview

The concept of this project is to showcase a diverse skillset demonstrating the ability to develop and integrate multiple parts of a platform. The choice to deliver an IoT concept however cliché was not made without thoughtful consideration of the utility derived from such a project in looking for work post-graduation. The following system overview provides a high-level description of the project.

1.2.1 Specifications

Data Transmission:

- Data will be moved from node to node using the MQTT (Message Queuing Telemetry Transport) Protocol.

Firmware:

- Firmware must Connect to a standard wireless network 802.11 b/g/n (at least one)
- Must implement a MQTT client (I must write this)
- Must support at least one action observable to the outside world
- Must use good coding practices and be well commented

Mobile Application:

- Must control observable action implemented in firmware
- Must use good coding practices and be well commented
- Must be easy to extend functionality
- Data and settings must persist across application uses.

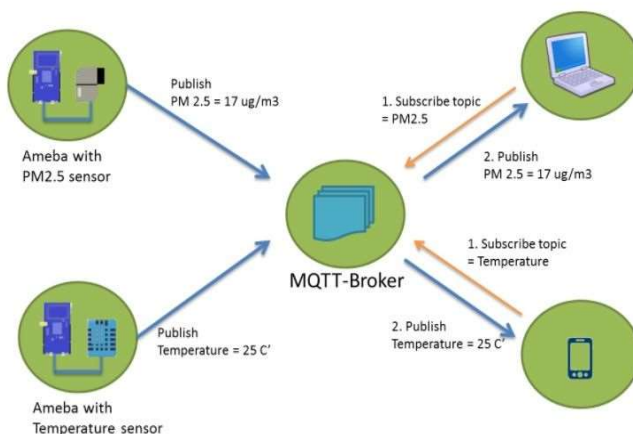
1.2.2 Summary of Core Components

Issued By:	Approved By:	Effective Date:	Page 8 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

The system is designed around the MQTT protocol version 3.1.1 published December 10th, 2015 by the OASIS (Organization for the Advancement of Structured Information Standards). MQTT was chosen for its flexibility and extensibility. Deploying a pub/sub messaging system, the MQTT protocol allows the entire system to be tolerant to lapses in connectivity. Because MQTT does not require persistent connectivity to function as a communication network, devices with power constraints can connect only periodically to update and be updated on system status. This is desirable for resource constrained environments such as embedded systems. Figure 1 illustrates the MQTT network architecture.

Figure 1 - MQTT Communication [1]



As seen in Figure 1, the broker manages subscriptions and publications keeping track of who wants what information. The project uses CloudMQTT.com as a broker service.

The project utilizes the Atmel SAMD21 Xplained pro development board with a WINC1500 2.4GHz WiFi module. This board has an ARM Cortex M0+ based microcontroller from Atmel's SAMD line. This hardware platform was chosen to remove the need to develop hardware on top of software as the given time-frame did not allow for both. This hardware platform also allowed for a completely integrated tool-chain during development dictating the development tool-chain, which also consisted of the Atmel Software Framework (ASF) and Atmel Studio for abstraction and an integrated development environment respectively.

The Atmel Software Framework's WiFi stack's use of call-back functions informed the decision to make the embedded MQTT client event driven. The call-back system keeps track client state triggering activities as required by the client or requested by incoming data. Put simply, the embedded client parses commands and manipulates the PWM output as instructed. A typical startup for the MQTT client might be to connect to the broker, subscribe to a topic representing its unique ID and wait for the WiFi stack to tell it new data is ready for processing. The operation of the MQTT client will be discussed at length later in this report.

Issued By:	Approved By:	Effective Date:	Page 9 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

The Android mobile application was written using the Xamarin Forms framework using .Net core. Its core function is to provide an easy to understand, configurable and extensible interface for the consumer. On first run, the user is presented with a settings screen to configure the application. When that is done, the user is taken to the add module page, where they are to configure a new module for use. Once this is done, the user can start manipulating the configured module. The mobile application uses the sqlite-net PCL library to store and recall data and allow data to persist across application starts. The OpenNetCF MQTT library was used to implement the MQTT client.

2. REFERENCED DOCUMENTS

Table 1: Reference Documents

Title	Revision Number	Comment
Project proposal	1.0	Submitted 8/27/2018
Functional Requirements	1.0	Submitted 8/27/2018
Test Plan	2.0	Submitted 8/27/2018
MQTT Specification	3.1.1	OASIS Standard

3. SYSTEM-WIDE DESIGN DECISIONS

Through research, it was determined the MQTT protocol was a perfect fit for the project, allowing the platform to be flexible and extensible in operation and deployment. Adopting the MQTT protocol meant the project could use industry standard HTTPS, SSL, and TLS secure communication models. While not a formal requirement, this provides a significant advantage in potentially bringing the platform to market.

3.1 Hardware

Having assessed the development board market, the SAMD21 Xplained Pro development platform from Atmel was chosen to facilitate firmware testing. This platform provides all hardware required to test IoT operations. Hardware components of this dev-board in use include SAMD21 ARM Cortex M0+ microcontroller, debugging/programming interface, FCC compliant WiFi module and some peripheral discrete devices to test PWM output with.

3.2 Firmware

Because of the erratic intervals is expected to enter the WiFi module, an event driven methodology was adopted for the MQTT client. A callback function was written to parse incoming data and hand actionable data off to the hardware interface layer of the PWM driver.

3.3 Android Application

Issued By:	Approved By:	Effective Date:	Page 10 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.: 0001	Version: 01

Internet of Things Proof of Concept

The mobile application targeted Android API levels 19 through 27. The following distribution indicates this interval would cover 96.5% of Android devices in the wild.

Figure 2 – Android API Level Distribution [2]

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.5%
4.3		18	0.4%
4.4	KitKat	19	7.6%
5.0	Lollipop	21	3.5%
5.1		22	14.4%
6.0	Marshmallow	23	21.3%
7.0	Nougat	24	18.1%
7.1		25	10.1%
8.0	Oreo	26	14.0%
8.1		27	7.5%

4. SYSTEM ARCHETECTURE

The following sections describe basic MQTT functionality and how payload data is compiled for transport. The design principles and interoperations associated with the Android application and firmware are discussed.

4.1 Overview of MQTT Protocol

The MQTT protocol will not be fully qualified in this document. Rather, the behaviors and functions germane to this project will be briefly discussed to provide context to deeper descriptions of developed code. The full MQTT protocol specification can be found at MQTT.org.

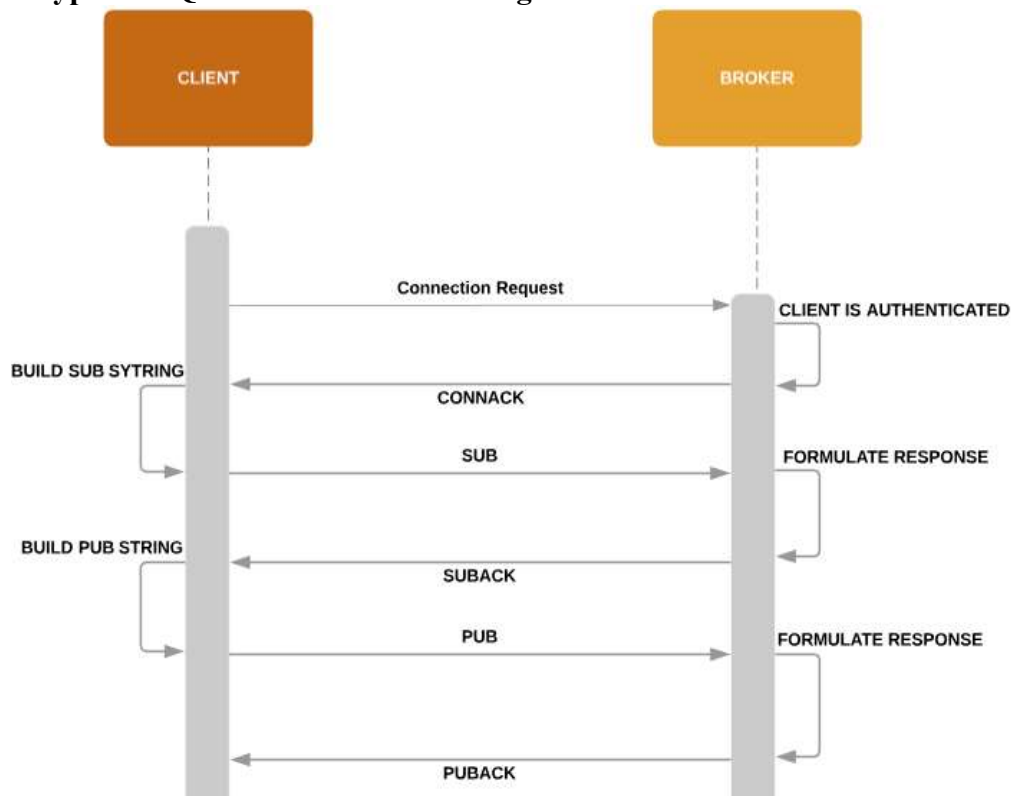
4.1.1 Connectivity

MQTT is a closed loop protocol. For every actionable packet, a response is expected. Either an ACK or Error state should be received to characterize the state of the connection or attempted communication. The following illustration describes a typical come up sequence.

Issued By:	Approved By:	Effective Date:	Page 11 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

Figure 3 - Typical MQTT Connection Exchange

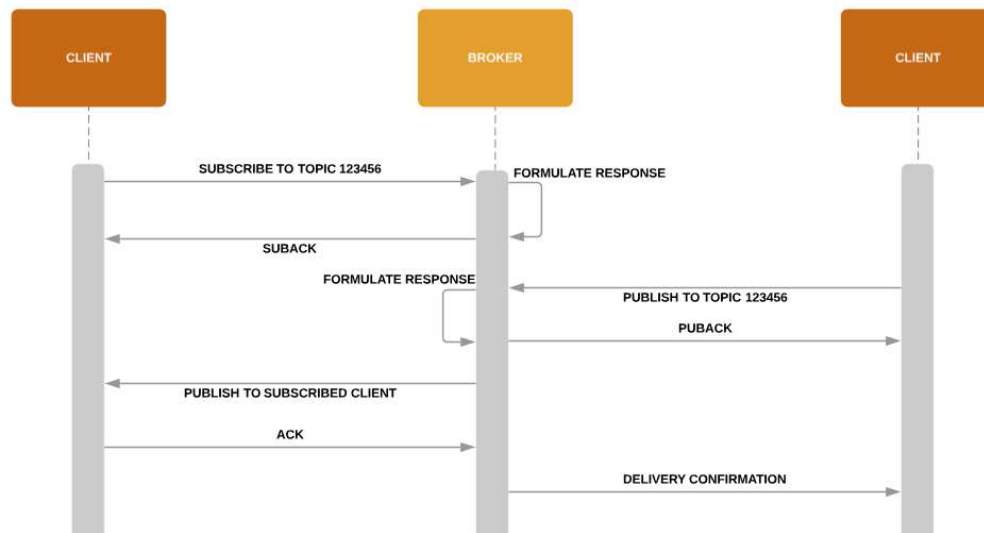


Following connection and after registering subscriptions and publishing any pending data, the client simply waits for the broker to send over new data. If the client is subscribed to a topic, and connected to the broker, any new publishes will be pushed down to all subscribed clients. Figure 4 illustrates one way this interaction could manifest.

Issued By:	Approved By:	Effective Date:	Page 12 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

Figure 4 - Pub/Sub Illustrated



4.1.2 Packet Structure

The MQTT packet structure consists of three parts: Fixed Header, Variable Header and Payload. The fixed header determines packet function indicating the remaining length of packet. Not all packet types use a Variable Header, but most do. The Variable Header contains a packet identifier unique to the session. Information in the Variable Header is determined by the control packet type indicated in the fixed header. Not all control packet types include a payload, but most do. Payload content (or if one exists at all) is also determined by the control packet type indicated in the fixed header. Tables 2 through 4 illustrate a common connection packet. An in-depth description of the anatomy of packet structure for the MQTT protocol can be found in the [OASIS MQTT 3.1.1 standard](#).

Table 2 Connection Fixed Header

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	Remaining Length								Packet Type				Reserved			
Data	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	0

Table 3 Connection Variable Header

Byte	9	8	7	6	5	4	3	2	1	0
Function	Keep Alive		Connect Flags	Protocol Level	Protocol Name				Protocol Name Length	
Data	0x00	0x80	0xCE	0x00	T	T	Q	M	0x04	0x00

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 13 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

Table 4 Connection Payload

Byte	9	8	7	6	5	4	3	2	1	0
Function	Data				Data Length		Data		Data Length	
Data	T	T	Q	M	0x04	0x00	Q	M	0x03	0x00

4.2 Payload Structure

The payload structure deployed for the project consists of command and a value. The command is a single char and the value is a formatted string. Table 5 details an example payload. Per protocol documentation, data length may be expressed as a single byte if two bytes are not required. The structure described below only applies to publishing data.

Table 5 Example Payload Format

Byte	5	4	3	2	1	0
Function	Value			length	Data	Length
Data	'3'	'2'	'1'	0x03	'1'	0x01

Numerical values are encoded as strings because the command char gets concatenated with the value. It is parsed apart when it arrives at the target device. Table 6 describes the available command codes and range of values accepted by the embedded MQTT client for PWM manipulation.

Note: This can be easily expanded to accept more command codes and more data-types. The table only represents what was chosen for this proof of concept.

Table 6 Accepted Commands

Char Value	Function
1	PWM
2	On/Off
Value	0 - 999

The function PWM would be paired with a value to set duty-cycle. On/Of would be paired with either a 1 or a 0, on and off respectively to turn the PWM peripheral on and off. Further discussion on specific methods of implementation will be discussed in the “Android Application”, “Firmware” and “Code Characterization” sections.

4.3 Android Application

The following sections describe theory of operation for the Android mobile application with its purpose being to describe application architecture. While code snippets are presented, readers should refer to the Code Characterization section for a full description of each functional block of code.

Issued By:	Approved By:	Effective Date:	Page 14 of
		10/31/2018	91
Travis Cucore	Travis Cucore	Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

4.3.1 Libraries in Use

To expedite development efforts, some libraries were used where their function was either duplicating functions written for the embedded platform, or their function was not germane to core IoT concepts.

4.3.1.1 sqlite-net-pcl (1.0.17253)

The sqlite-net-pcl library is available on the Microsoft NuGet package management system. This library provides utility methods to abstract database interactions. This package uses the Linq library provided in the .NET framework to enable sql-like querying to an object attached to the actual sql tables.

4.3.1.2 OpenNETCF.MQTT (1.4.118)

The OpenNETCF.MQTT library is available on the Microsoft NuGet package management system. This library provides a fully functional MQTT client. While a client could have been written from scratch, a client had already been written for the embedded portion of the project which made writing this client redundant from an experiential view.

4.3.1.3 Xamarin.Forms (2.5.0.122203)

The Xamarin.Forms library provides a framework for developing cross-platform mobile applications. This framework was chosen because of its use of Portable Class Libraries (PCL) allowing a single code-base to be deployed across multiple platforms with minimal additional coding. While the project does not present an IOS or Windows Mobile application, the Xamarin.Forms framework makes it easier to port to those platforms if commercialization is desired.

4.3.2 Classes

The following sections describe each functional class type and its role in the mobile application. Detailed descriptions of each functional block of code can be found in the Code Characterization section.

4.3.2.1 Data

For this report, data classes are classed that represent a specific type of data. These classes are used to populate sql tables and GUI attributes while keeping track of system status and application settings and preferences. There are three types of data classes.

AppSettings.cs – Defines a class for storing application settings. Contains a primary key associated with the app settings database and all fields required to setup the application for use with a broker. A field for default location is also present in this class. All fields are reachable via get/set. Uses SQLite library

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 15 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

Module.cs – Defines a class for storing module data. Contains a primary key associated with the ID field which automatically increments when a new object is created. Ultimately this class should contain several class extensions to characterize different types of modules. All fields are reachable via get/set. Uses SQLite library

Profile.cs – Not yet implemented. Defines a class for storing system module states or a subset of system module states. Includes a primary key which is automatically incremented when a new object is created. Provides a file for storing a list of module states that is accessible via get/set. Uses System.Collection.Generic and SQLite libraries.

4.3.2.2 Database

For this report, database classes are classes are wrappers for the SQLite library in use. There are three database classes, one for each type of data: AppSettingsDatabase.cs, ModuleDatabs.cs and ProfilesDatabase.cs. These classes are identical in their operations. For this reason, only one database class is described.

ModuleDatabase.cs – Defines a class to function as a wrapper for the SQLite library. This class is instantiated as an object when the application is started. Contains asynchronous tasks to perform typical database operations including:

- Creating a new database
- Creating a database table
- Retrieving a database table
- Retrieving a database entry by primary key
- Saving an entry
- Deleting an entry

This class uses the following Libraries:

- SQLite
- System.Collections.Generic
- System.Threading.Tasks

4.3.2.3 MQtil.cs

MQtil.cs is a wrapper for the MQTT client. The main purpose of this class is to allow for multiple client connections and some abstraction, compiling publish strings, and debugging feedback. Wrapping this library also allows for multiple client connections. While not implemented today, it would be possible to interface with several brokers from the same application instance. This class is instantiated once when the application starts and may be destroyed then recreated as needed depending on connection status. This is a byproduct of doze-mode, an Android power management function to be discussed in section 4.3.6.

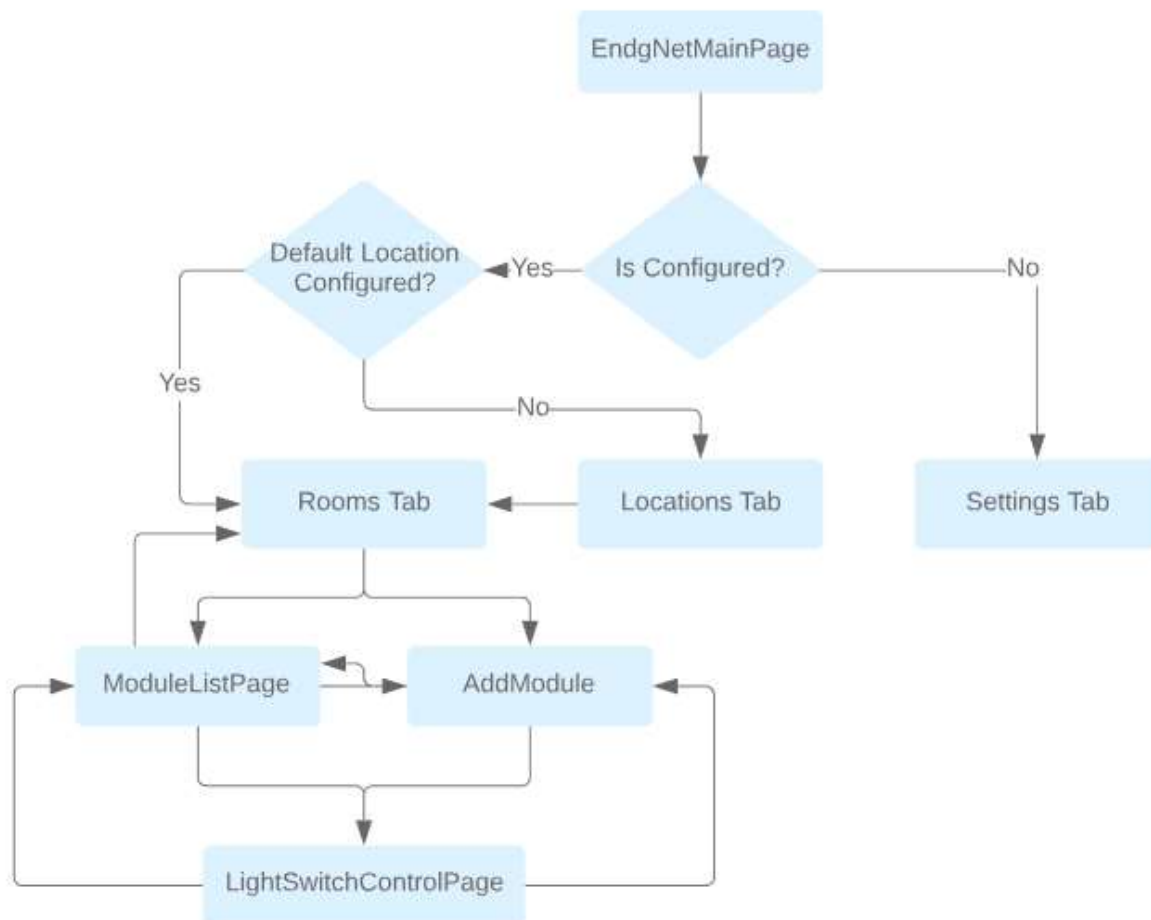
Issued By:	Approved By:	Effective Date: 10/31/2018	Page 16 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

4.3.3 Views

There are several views serving various purposes. This section will illustrate how data moves from one view to another and the function each view serves. A more detailed discussion of the anatomy of this code can be found in the Code Characterization section of this document. Figure 5 describes navigation between views in the EdgeNet application.

Figure 5 - EdgeNet Navigation Tree



Issued By:	Approved By:	Effective Date:	Page 17 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

4.3.3.1 AddModule

The AddModule View is used to add new modules to the list of modules recognized by the EdgeNet application.

Figure 6 - AddModule View (New Module)

This view is also used to edit an existing module. When used for this purpose, module data is populated, and it becomes possible to delete the module being edited. Figure – 7 illustrates this version of the AddModule view.

Figure 7 - AddModule View (Edit Module)

When called, the AddModule view accepts two arguments of type bool and Module representing if the module is new and data associated with the module to be edited respectively. The Module argument may not be left blank but may be passed NULL if adding a new module as there will

Issued By:	Approved By:	Effective Date:	Page 18 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

be no existing data to pass along. This view is also data-bound using binding context and XAML to populate relevant data in the user interface. This data-binding also makes sure the bound object always represents data found in the bound UI object. Figures 7 and 8 Illustrates how this view is called in both cases.

Figure 8 - Calling AddModule View (New Module)

```
//push add module page and set binding context
await Navigation.PushAsync(new AddModule(true, null) { BindingContext = new Module() });
```

Note, the binding context is set by creating a new Module object. This provides a place to store the information about to be captured from the user. Also notice the first argument is set to true and the second set to null. This tells the view its purpose is to create a new module and there is not module data to be associated with the binding context.

Figure 9 - Calling AddModule View (Edit Module)

```
await Navigation.PushAsync(new AddModule(false, module) {BindingContext = module });
```

In this case, the first argument is set to false while the second argument is set to a Module object instance. Additionally, the binding context is also set the module object instance. This tells the AddModule View that this is not a new module and provides the view with the information required to bind to the Module instance object to be edited. This saves a lot of programmatic effort.

4.3.3.2 AddProfile

The add profile view is not written, as such, it is not implemented. Profile management was something being toyed with during development, but not essential to the project. It is mentioned because it exists in this document because it exists in the solution as a view.

4.3.3.3 EdgeNetMainPage

EdgeNetMainPage is a tabbed view serving as the main user interface for the application consisting of the following tabs: The following actions take place when the application is loaded, or when the view is brought back into focus:

1. The selected item (if any are selected) is deselected and the current tab is set to rooms tab.
2. An attempt is made to get application settings from a database. If no database exists, one is created.
3. An attempt is made to get module data from a database. If no database exists, one is created.
4. An attempt is made to get profiles from a database. If no database exists, one is created.
5. A list of distinct locations is set as the item source of the locations tab

Issued By:	Approved By:	Effective Date:	Page 19 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

6. A check for current settings is performed.
 - a. If no settings exist (empty table or content = null) the user is brought to the settings tab to configure the application after being notified the broker has not been configured.
 - b. If settings exist, the MQTT client is configured and connected to the broker, but only if the client is not already connected to a broker.

Settings – This tab provides the interface required to configure the application to work with a broker and set a default location. It is data-bound to current application settings.

Figure 10 - Settings Tab

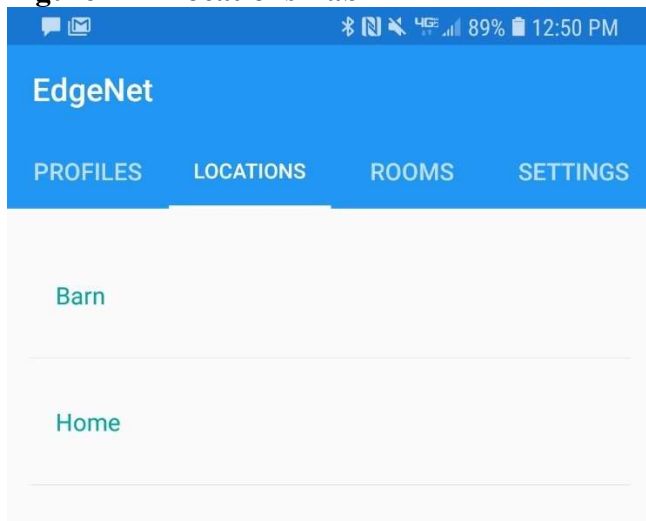
The screenshot shows the 'Settings' tab in the EdgeNet application. The interface has a blue header with the 'EdgeNet' logo and an 'EDIT SETTINGS' button. Below the header are four tabs: 'PROFILES', 'LOCATIONS', 'ROOMS', and 'SETTINGS', with 'SETTINGS' being the active tab. The main content area contains several input fields: a text field with 'm12.cloudmqtt.com', a text field with '14500', a text field with 'admin', two password fields (indicated by dots), and a text field with 'Home'. At the bottom of the form is a grey button labeled 'SAVE SETTINGS'.

Locations – This tab provides a sorted list of locations with configured modules recognized by the application.

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 20 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

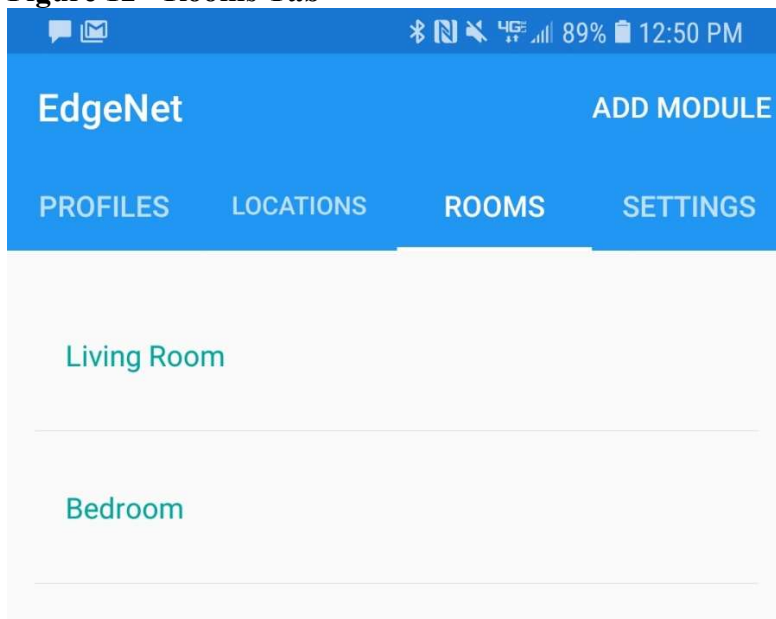
Internet of Things Proof of Concept

Figure 11 - Locations Tab



Rooms – This tab provides a sorted list of rooms associated with either the location selected from the Locations tab. This tab will automatically populate with rooms associated with the default location upon application startup if a default location is set.

Figure 12 - Rooms Tab



Profiles – This tab is not functional. The idea of profiles was toyed with during development but did not get off the ground in time to be fully operational. It exists in this document because the tab is visible in the application but does not serve a purpose.

Issued By:	Approved By:	Effective Date:	Page 21 of 91
Travis Cuore	Travis Cuore	10/31/2018	
		Document No.: 0001	Version: 01

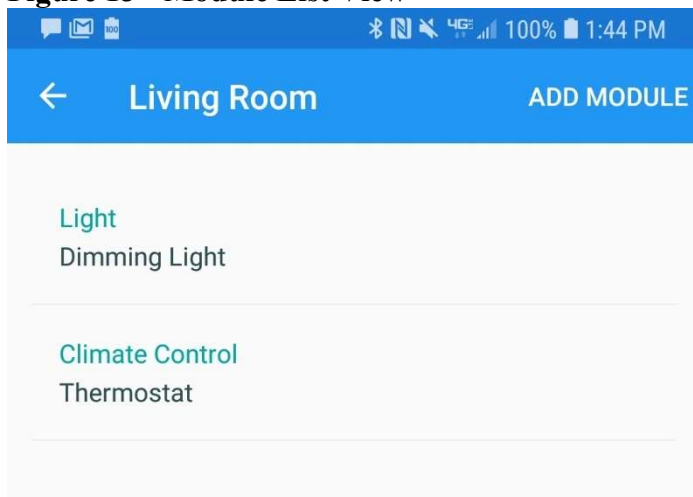
Internet of Things Proof of Concept

Note: The EdgeNetMainPage view also contains all code-behind for each tab.

4.3.3.4 ModuleListPage

The ModuleListPage view presents the user with a list of modules associated with the room selected. Each module is identified by name and type.

Figure 13 - Module List View



Selecting a module will open the LightSwitchControlPage, passing the appropriate information to it and binding the UI to that data.

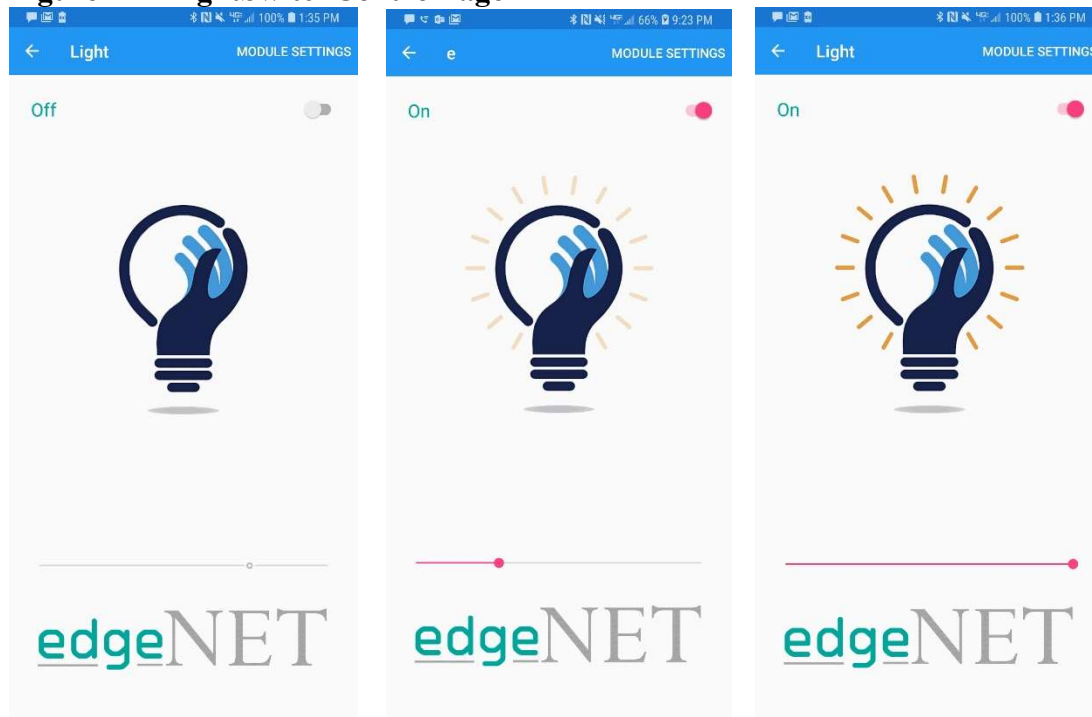
4.3.3.5 LightSwitchControlPage

This view provides the ability to turn a light on or off and adjust the brightness of the light. The current status of the light is fed back to the user visually using an image that represents the lights state and brightness as shown below.

Issued By:	Approved By:	Effective Date:	Page 22 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

Figure 14 - LightSwitchControlPage



This view accepts one argument of the type Module used to pass in information about the module being manipulated. This information is updated as the user interacts with the view. On disappearing, module data is saved back to the database.

4.3.3.6 MainPage

While this view exists in the solution, it is left empty. This was the default view called from App.cs on start. It cannot be removed and exists in this document only because it exists in the views folder in the solution explorer.

4.3.4 Localized Interfaces

To develop portable code capable of compiling for multiple platforms, some tasks must be localized to each target platform. EdgeNet uses local resources to store and access application data. To enable this functionality, localized interfaces were developed. The local interface used to work with sql data consists of two parts. A base-class is written to provide a bridge between portable code and code to be compiled for a specific platform. These are linked with assembly references. A class extension is written into the localized solution as an interface. For EdgeNet, these files are called ILocalFileHelper.cs and LocalFileHelper.cs respectively.

Issued By:	Approved By:	Effective Date:	Page 23 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

The base class defines a function prototype allowing the function name to be called from shared code.

Figure 15 - Shared Code Interface

```
namespace EdgeNet
{
    6 references
    public interface ILocalFileHelper
    {
        5 references
        string GetLocalFilePath(string fileName);
    }
}
```

The localized class extension will contain code doing work.

Figure 16 - Localized Code Interface Class Extension

```
public class LocalFileHelper : ILocalFileHelper
{
    0 references
    public string GetLocalFilePath(string fileName)
    {
        string docFolder = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        string libFolder = Path.Combine(docFolder, "..", "Library", "Databases");
        if(!Directory.Exists(libFolder))
        {
            Directory.CreateDirectory(libFolder);
        }
        return Path.Combine(libFolder, fileName);
    }
}
```

The localized class extension is mapped to the shared code class by assembly reference. This assembly reference tells the compiler how to stitch the two together.

Figure 17 - Localized Interface Assembly Reference

```
[assembly: Dependency(typeof(LocalFileHelper))]
```

4.3.5 Data Bindings

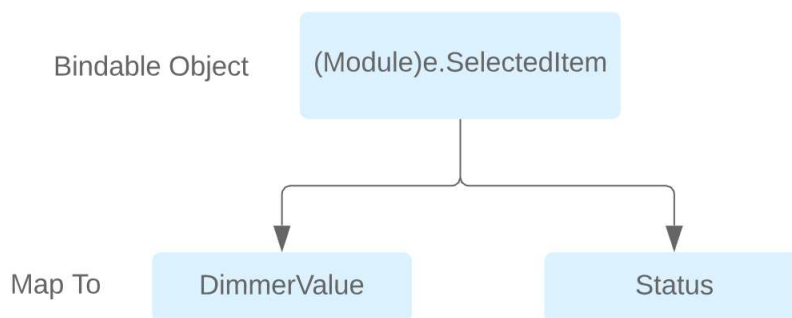
Data-bindings have been glossed over in earlier sections. This section will clearly describe what a data-binding is, and how it works.

Issued By:	Approved By:	Effective Date:	Page 24 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.: 0001	Version: 01

Internet of Things Proof of Concept

Data-bindings are used to map data in an object to data in another object. For the EdgeNet Android application, this means mapping Module data with UI properties. This can be done in XAML or Code-Behind. The EdgeNet application uses XAML for mapping data-bindings and code-behind to pass around bindable objects.

Figure 18 - Map to Bindable Object



In the EdgeNet application, bindable objects are passed to a view when it is created or called along with a reference to the object for binding context.

Figure 19 - Calling View with Binding Context

```
//push the module interface page and pass data for the selected item
await Navigation.PushAsync(new LightSwitchControlPage((Module)e.SelectedItem) { BindingContext = (Module)e.SelectedItem});
```

In the EdgeNet application, mapping UI properties to properties of the bindable object is done in XAML.

Figure 20 - Binding Properties to UI Elements

```
<Slider x:Name="dimmerSlider" ValueChanged="OnSlider_ValueChanged" Value="{Binding DimmerValue}" Minimum="0.0" Maximum="999" />
```

This creates a bi-directional link between the bound object property and the bound property of the UI element. In this example, the Value of the slider is mapped to the DimmerValue property of the module being manipulated.

4.3.6 Doze Mode

Doze mode is a power management feature of the Android operating system rolled out in API version 23. What doze mode does, is turn off the screen and antennas a short period of time after the phone has not been interacted with while leaving the application to run. Doze mode does allow for applications to get data out during regularly scheduled maintenance windows wherein the antennas are briefly turned back on. However, these windows get increasingly far apart and does not announce itself to the application.

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 25 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

This previously unknown behavior impacts connectivity to the MQTT broker which requires a keep-alive every so often to consider the connection valid.

There is one way around doze-mode, which is to ask the user for a wake-lock, which allows the application to prevent the device from going into doze-mode. Unfortunately, there are very few instances where Android considers a wake-lock acceptable. An application implementing wake-lock without an approved reason will be removed from the Android market. This mode of operation can be worked around by using Android Firebase Cloud Messaging service instead of MQTT or writing an MQTT client to comply with its behavior. Either would require the application to be rearchitected meaning those bugs that do exist around connectivity will remain in place due to time constraints for senior design.

4.4 Firmware

The firmware deployed for this project was developed using the Atmel Software Framework and a canned WiFi stack provided as a bolt on to that framework. The standard hardware Abstraction and Interrace layers were used where possible to reduce development efforts as they are part of the Atmel toolchain. The entire firmware package is built on C-language and compiled to assembly using the Atmel Studio IDE, which is built on visual studio.

The following standard C-language libraries are in use for this project.

- **Stdlib.h**
 - This library provides definitions for common types, variables, and functions.
- **Stdio.h**
 - This library provides the frameworks required to use serial debugging.

4.4.1 Atmel Software Framework

The Atmel Software Framework (ASF) is a collection of libraries and abstractions provided by Atmel as part of their embedded toolchain to make embedded development easier. This project makes use of the following parts of the ASF by including them as headers.

- **Asf.h**
 - This header provides a list of includes required for the platform being worked on. It is automatically generated when a project is started for a specific platform.
- **Driver/include/m2m_wifi.h**
 - This header is required to make use of the WiFi stack header provided by Atmel for the WINC1500 WiFi module.
- **Socket/include/socket.h**
 - This header provides an interface required for socketed network connections.
- **Samd21j18a.h**

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 26 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

- Provides hardware abstraction layer required to write hardware interface layer for PWM driver.

4.4.2 WiFi Stack

The WiFi stack is fully functional when selected as part of the build when the project is started providing a callback function to interface with. Its purpose is to manage connection to the AP providing internet access. Two parts of the WiFi stack are exposed to the developer outside calling functions directly from the driver. The first is the WiFi callback function and the second is the Socket callback function. The socket callback function is what the MQTT client ties into as it provides status relating to socket connection, message received and message sent. This integration will be described in section 4.4.6 along with the anatomy of the MQTT client.

4.4.3 Abstractions

This section will only cover abstractions provided by the ASF as abstractions used for the PWM driver and the MQTT client are described in their respective sections.

Abstractions provided in the form of #defines are provided for the WiFi stack. These abstractions are used to setup connection to networks and hosts.

- **#define MAIN_WLAN_SSID** – defines the SSID the module should connect to.
 - = “Cucore_AdHoc”
- **#define MAIN_WLAN_AUTH** – defines what form of authentication if any should be expected when connecting to the SSID defined above.
 - = M2M_WIFI_SEC_WPA_PSK – This is an enumeration defined in another header.
- **#define MAIN_WLAN_PSK** – defines the passkey used to connect to SSID defined above if authentication is required
 - = “*****”
- **#define MAIN_HOST_NAME** – defines hat host to connect to for SSL connection on startup
 - = “m12.cloudmqtt.com”
- **#define MAIN_HOST_PORT** – defines what port to use when connecting to host defined above.
 - = 24500

Finally, there is an enumeration for socket status, which doesn’t get used by the MQTT client or PWM driver but is used by the socket library to manage the SSL socket.

4.4.4 PWM Driver

Issued By:	Approved By:	Effective Date:	Page 27 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

The PWM driver consists of two files, PWM.c and PWM.h. The header only contains function prototypes. The .c file contains all setup routines and the hardware interface layer (HIF). This section will describe the basic operation and construction of this drive reserving more detailed information for the Code Characterization section.

4.4.4.1 Libraries in Use

The only library in use for this driver is the samd21j18a.h file which provides the hardware abstraction layer (HAL) as well as access to many standard libraries as part of the Atmel Software Framework. The HAL definitions provided in samd21j18a.h including libraries includes therein are used to construct the HIF as well as the setup routine for the PWM driver.

Of the includes encapsulated in the samd21j18a.h file, the following are used in this driver:

- **Components/port.h** – Contains abstractions for port-pins
- **Components/tc.h** – Contains abstractions for timer counters
- **Components/tcc.h** – contains abstractions for timer counter control registers

4.4.4.2 Peripheral Configuration and Initialization

The PWM peripheral configuration and initialization sequence is encapsulated in the StartPWM() function as part of PWM.c. The process required to bring up a PWM peripheral is rigid leaving little to no room for creative influence. This means, just about every PWM bring up routine will look the same for the chip, IP spec, and architecture used in this project. In fact, formal documentation specifies what order things need to happen in order for a successful bring up of the PWM peripheral.

For the SAMd21 ARM Cortex M0+, getting the PWM peripheral setup is a little more complicated than it would be for a memory mapped microcontroller. That is because the peripherals are not memory mapped, and each pin may have several peripherals attached to it via multiplexer. The following sequence was followed to get the PWM peripheral running for this project.

1. Configure port pin direction and control register
2. Attach port pin to multiplexer and set to output pin
3. Power on the timer counter device
4. Wait for device to synchronize
5. Set timer counter control register to count up
6. Wait for device to synchronize
7. Tell peripheral device to operate in NPWM mode (normal pulse-width-modulation)
8. Wait for device to synchronize
9. Set period for PWM to 999 ticks
10. Wait for synchronization

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 28 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

At this point the PWM signal is set but is not turned on because the desired default state is off. When called upon by the MQTT client, the hardware interface layer will enable the timer counter using the timer counter control register. This operation is discussed in the next section. The code that makes all this happen will be fully described in the Code Characterization section of this document.

4.4.4.3 Hardware Interface Layer

The hardware interface layer (HIL) was developed as a part of this project and consists of 3 functions. Their function prototypes and functional descriptions are found below. These functions are fully characterized in the Code Characterization section.

void SetBrightness(unsigned int)

This function sets the duty-cycle of the PWM peripheral by applying a value between 1 and 999 to the counter control buffer (CCB). The counter control buffer will commit the new duty-cycle to the counter control register (CC) during the next synchronization. Attempting to alter the duty-cycle outside of synchronization can cause firmware to perform unpredictably. After setting the duty-cycle, the timer counter is enabled in the time counter controller register.

void LEDOff(void)

This function disables the PWM peripheral by clearing the timer counter enable bit in the timer counter control register.

void LEDOn(unsigned int)

This function enables the PWM peripheral by setting the timer counter enable bit in the timer counter control register.

4.4.5 MQTT Client

The MQTT client was developed to support basic functions but left doors open to develop a more robust and fault tolerant client. This section describes the libraries and abstractions in use, then illustrates how data flows through the client. While some code snippets may be called out here, a full characterization of the MQTT clients code can be found in the Code Characterization section of this document.

4.4.5.1 Libraries in Use

The following libraries are included as headers in the MQTT.c file supporting its function.

- **Stdlib.h** – This is the standard C-language library. It contains standard definitions and macros and encapsulates other common headers.
- **Stdio.h** – This is the standard I/O library provided by C. This library is required to enable serial debugging during device operation.

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 29 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

- **Driver/include/mdm_wifi.h** – This library contains the hardware interface and abstraction layers for the WiFi module in use. This library is required to make use of socket abstractions and structures.
- **Memory.h** – This library is required to use atomic read/write operations when allocating and clearing large buffers to big for primitive data-types.
- **PWM.h** – Provides hardware function prototypes for hardware interface layer to PWM peripheral.

4.4.5.2 Abstractions

MQTT.h contains several abstractions for the MQTT client. Most of these abstractions exist only to make code more readable. This section provides a brief overview of abstraction types found in this file as well as one structure used to parse incoming data.

Fixed Header Abstractions – This section of the file contains #defines used to make compiling a fixed header more human readable. These abstractions include: Control packet type and fixed header flags.

```
// Fixed Header Control Packet Types (byte 1)
#define CONNECT      1
#define CONNACK      2
#define PUBLISH      3
#define PUBACK       4
#define PUBREC       5
#define PUBREL       6
#define PUBCOMP      7
#define SUBSCRIBE     8
#define SUBACK        9
#define UNSUBSCRIBE  10
#define UNSUBACK      11
#define PINGREQ       12
#define PINGRESP      13
#define DISCONNECT    14
#define Reserved      15 //This is an invalid packet type, do not use it

//#defines for manipulating fixed header flags
#define FH_RETAIN(n)  n<<0
#define FH_QOS(n)     n<<1
#define FH_DUP(n)     n<<3
#define PACKET_TYPE(n) n<<4
```

Issued By:	Approved By:	Effective Date:	Page 30 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

Variable Header Abstractions – This section of the file contains #defines used to abstract Variable Header flags.

```
//#defines for Variable Header flags
#define VH_CLEAN_SESSION(n)  n<<1
#define VH_WILL_FLAG(n)      n<<2
#define VH_QOS(n)             n<<3
#define VH_WILL_RETAIN(n)     n<<5
#define VH_PASS(n)            n<<6
#define VH_USER_NAME(n)       n<<7
```

Callback Abstractions – This section of the file contains #defines related to MQTT connections. Specifically, all possible states of connection and responses to those states are represented. This section also defines abstractions for pub/sub actions and responses.

```
//MQTT: Callback defines related to connection requests
#define CONNACK_CB          0x20
#define ACCEPTED            0
#define BAD_PROTOCOL_VERSION 1
#define ID_REJECTED         2
#define SERVER_UNAVAILABLE  3
#define INVALID_CREDENTIALS 4
#define NOT_AUTHORIZED      5
#define SESSION_PRESENT     0
```

```
//MQTT: Callback defines related to publish requests
#define PUBACK_CB          0x40
```

```
//MQTT: Callback defines related to subscription requests
#define SUBACK_CB          0x90
#define SUCCESS_QOS0       0x00
#define SUCCESS_QOS1       0x01
#define SUCCESS_QOS2       0x02
#define SUBACK_FAILED      0x80
```

```
//MQTT: Callback defines related to received messages
#define MESSAGE_RECEIVED    0x30
```

Constants – This section defines constants needed to build a connection string, initialize a buffer for MQTT to work with and limit the maximum size of a MQTT message.

Issued By:	Approved By:	Effective Date:	Page 31 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.: 0001	Version: 01

Internet of Things Proof of Concept

//Constants... These will not change.

```
#define PROTOCOL_NAME    "MQTT" //This should not change according to 3.1.1 MQTT
specification
#define PROTOCOL_LEVEL    4 //set protocol level here default is 3.1.1
#define KEEP_ALIVE        0 //default keep alive is 0 (do not disconnect due to inactivity)
#define BUFFER_SIZE        1024 //Size of MQTT Buffer
#define RECV_TIMEOUT        0 //Use this in recv() for timeout. 0 indicates no timeout
(default)
#define MAX_MESSAGE_SIZE    64 //maximum number of bytes a message can contain
(inbound or outbound)
```

Quality of Service Abstractions – This section defines quality of service levels used to tell the broker how a published string should be handled.

//QOS abstractions

```
#define ONCE_AT_MOST    0 //deliver the message up to one time
#define ONCE_AT_LEAST    1 //deliver the message at least one time
#define ONCE_EXACTLY    2 //deliver the message exactly one time
```

MQTT Command Abstractions – This section defines command types and some command data abstractions.

//MQTT CMD Type Abstractions

```
#define PWM    '1'
#define TOGGLE '2'
#define ON    '1'
#define OFF    '0'
#define STATUS '3'
```

4.4.5.3 ASF Integrations

The MQTT client integrates with the socket callback function provided by the WiFi stack.

The MQTT_Connect() function is called when a callback indicating the device has connectivity to an AP. Once a connection to the broker is made, the system is told to wait for an incoming message.

```
case SOCKET_MSG_CONNECT:
{
    tstrSocketConnectMsg *pstrConnect = (tstrSocketConnectMsg *)pvMsg;
    if (pstrConnect && pstrConnect->s8Error >= 0)
    {
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 32 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
printf("socket_cb: connect success!\r\n");
```

```
Mqtt_Connect(sock); //connect to broker
```

```
recv(sock, pvMsg, BUFFER_SIZE, RECV_TIMEOUT); //wait for ack
```

```

}
else
{
    printf("socket_cb: connect error!\r\n");
    close(tcp_client_socket);
    tcp_client_socket = -1;
}

```

When a message is received, the socket message received case of the socket callback function is activated. In this case the callback case receives the data into a buffer on chip and calls the MQTT callback function.

case SOCKET_MSG_RECV:

```

{
    tstrSocketRecvMsg *pstrRecv = (tstrSocketRecvMsg *)pvMsg;
    if (pstrRecv && pstrRecv->s16BufferSize > 0)
    {
        printf("socket_cb: recv success!\r\n");
        recv(sock, pstrRecv->pu8Buffer, BUFFER_SIZE, RECV_TIMEOUT);
        Mqtt_Callback(sock, pstrRecv);
    }
    else
    {
        printf("socket_cb: recv error!\r\n");
        close(sock);
    }
}

```

The next section will describe how data flows through the MQTT client.

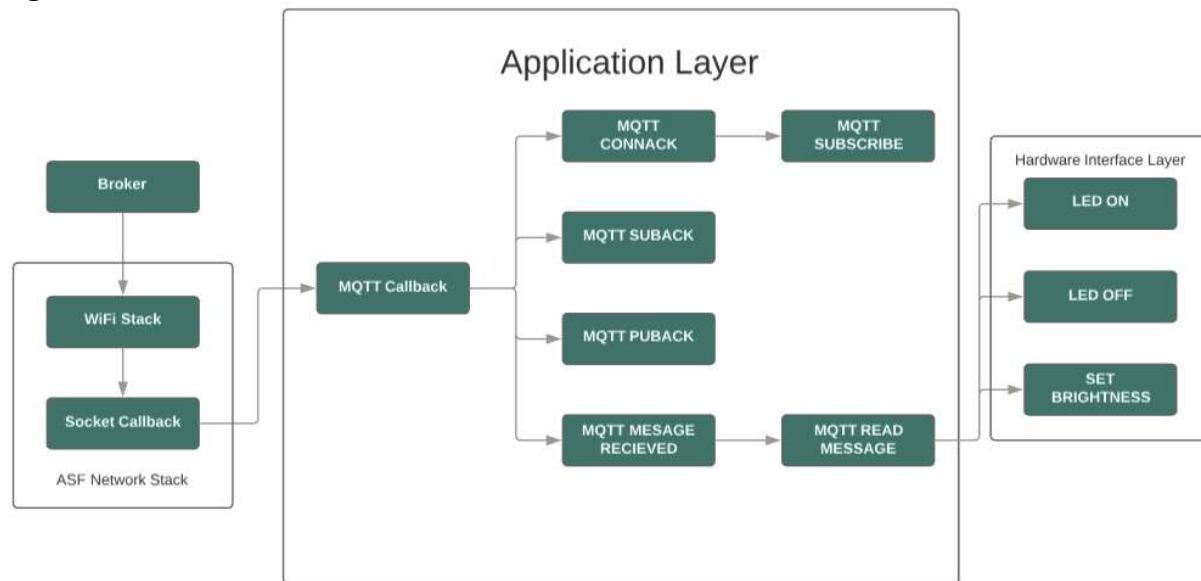
4.4.5.4 Data Flow

Data flows from the broker into the ASF WiFi stack resulting in a socket callback indicating there is new data to be received from the WINC1500 module. When this happens, data is pulled into the MCU buffer allocated for this data and passed to the MQTT callback function. The MQTT callback function determines the nature of this data by inspection of data passed to it and executes the correct action.

Issued By:	Approved By:	Effective Date:	Page 33 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

Figure 21 - Data Flow



5. CODE CHARACTERIZATION

The purpose of this section is to fully explain every line of code written for this project that is not immediately obvious given context and in-code comments. This means code-base considered part of the toolchain will not be discussed except where required to explain how code developed specifically for this project functions.

For each file written for this project, all lines of code will be presented as a whole or in smaller blocks and discussed in as much detail as is required to fully understand how it works.

5.1 Firmware

This section relies heavily on comments made in code at the time it was written to characterize what is going on. When code comments are insufficient to describe the intended function of the code and/or how a block of code accomplishes its task, further discussion will take place before it is presented, or the block of code will be broken up into smaller parts for discussion.

5.1.1 MCQTT.h

The socket header is included here to ensure abstractions associated with it are provided to the application code. The application code makes use of socket information when sending data to the broker.

```
#include "socket/include/socket.h"
```

Issued By:	Approved By:	Effective Date:	Page 34 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

Fixed header control packet types are defined by the MQTT protocol. These abstractions are purely to make code easier to read and understand.

```
// Fixed Header Control Packet Types (byte 1)
#define CONNECT      1
#define CONNACK      2
#define PUBLISH      3
#define PUBACK       4
#define PUBREC       5
#define PUBREL       6
#define PUBCOMP      7
#define SUBSCRIBE    8
#define SUBACK       9
#define UNSUBSCRIBE 10
#define UNSUBACK     11
#define PINGREQ      12
#define PINGRESP     13
#define DISCONNECT   14
#define Reserved     15 //This is an invalid packet type, do not use it
```

The following macros are used to set and clear flags for the fixed header. Table 7 characterizes the value types and ranges expected by each macro:

Table 7 - Fixed Header Macros

Macro Name	Values Accepted	Short Description
FH_RETAIN	TRUE/FALSE (1/0)	Retain tells the broker if it should hold on to the message once it's been delivered
FH_QOS	INTEGER BETWEEN 0 AND 2 INCLUSIVE	Quality of service tells the broker how to handle delivery.
FH_DUP	TRUE/FALSE (1/0)	Duplicate tells the broker if the message is a duplicate.
PACKET_TYPE	INTEGER BETWEEN 1 AND 14 INCLUSIVE	Packet type tells the broker what type of control packet this is.

```
//defines for manipulating fixed header flags
#define FH_RETAIN(n)  n<<0
#define FH_QOS(n)     n<<1
```

Issued By:	Approved By:	Effective Date:	Page 35 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
#define FH_DUP(n)      n<<3
#define PACKET_TYPE(n) n<<4
```

The macros are used to set Variable Header control flags. The following table characterizes the value types and ranges expected by each macro.

Table 8 - Variable Header Macros

Macro Name	Values Accepted	Short Description
VH_CLEAN_SESSION	TRUE/FALSE (1/0)	Tells the broker if the connection represents a clean session.
VH_WILL_FLAG	TRUE/FALSE (1/0)	DEPRECATED
VH_QOS	INTEGER BETWEEN 0 AND 2 INCLUSIVE	Tells the broker how to deliver the message.
VH_WILL_RETAIN	TRUE/FALSE (1/0)	Tells the broker if it should retain the message or not.
VH_PASS	TRUE/FALSE (1/0)	Tells the broker if the connection string includes a password.
VH_USER_NAME	TRUE/FALSE (1/0)	Tells the broker if the connection string includes a username.

```
///defines for Variable Header flags
#define VH_CLEAN_SESSION(n) n<<1
#define VH_WILL_FLAG(n)    n<<2
#define VH_QOS(n)          n<<3
#define VH_WILL_RETAIN(n)  n<<5
#define VH_PASS(n)         n<<6
#define VH_USER_NAME(n)    n<<7
```

The following abstractions are only used to make code more readable and supportable. These are all related to connection requests and the responses possible to a connection request.

```
///MQTT: Callback defines related to connection requests
#define CONNACK_CB      0x20
#define ACCEPTED        0
#define BAD_PROTOCOL_VERSION 1
#define ID_REJECTED     2
#define SERVER_UNAVAILABLE 3
```

Issued By:	Approved By:	Effective Date:	Page 36 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

```
#define INVALID_CREDENTIALS    4
#define NOT_AUTHORIZED        5
#define SESSION_PRESENT        0
```

The following abstraction represents the response expected from a broker to a publish request.

```
//MQTT: Callback defines related to publish requests
#define PUBACK_CB              0x40
```

The following abstractions represent possible responses from the broker to a subscription request.

```
//MQTT: Callback defines related to subscription requests
#define SUBACK_CB              0x90
#define SUCCESS_QOS0           0x00
#define SUCCESS_QOS1           0x01
#define SUCCESS_QOS2           0x02
#define SUBACK_FAILED           0x80
```

The following abstraction represents the value the MQTT callback block expects when a message is received.

```
//MQTT: Callback defines related to received messages
#define MESSAGE_RECEIVED        0x30
```

The following constants make up the configuration block for critical MQTT client settings. The following table describes the function of each abstraction.

Table 9 - Constants

Abstraction Name	What it does
PROTOCOL_NAME	Protocol name defines the protocol name property in the MQTT fixed header
PROTOCOL_LEVEL	Protocol level defines the protocol level value in the fixed header
KEEP_ALIVE	Keep-alive defines the passage of time in seconds required to disconnect due to inactivity. This is part of the connection packet.
BUFFER_SIZE	Buffer size is the value used to allocate memory for incoming data and to build strings for all MQTT activities.

Issued By:	Approved By:	Effective Date:	Page 37 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

RECV_TIMEOUT	Receive timeout is used to indicate how long the socket interface should wait before triggering a timeout when told to receive data.
MAX_MESSAGE_SIZE	The maximum message size that can be sent to the broker as a payload.

//Constants... These will not change.

```
#define PROTOCOL_NAME    "MQTT" //This should not change according to 3.1.1 MQTT
specification
#define PROTOCOL_LEVEL    4 //set protocol level here default is 3.1.1
#define KEEP_ALIVE        0 //default keep alive is 0 (do not disconnect due to inactivity)
#define BUFFER_SIZE        1024 //Size of MQTT Buffer
#define RECV_TIMEOUT        0 //Use this in recv() for timeout. 0 indicates no timeout
(default)
#define MAX_MESSAGE_SIZE 64 //maximum number of bytes a message can contain
(inbound or outbound)
```

These abstractions are used with the fixed and Variable Header QOS macros described earlier.

//QOS abstractions

```
#define ONCE_AT_MOST 0 //deliver the message up to one time
#define ONCE_AT_LEAST 1 //deliver the message at least one time
#define ONCE_EXACTLY 2 //deliver the message exactly one time
```

//abstractions to make binary arguments more human readable

```
#define TRUE 1
#define FALSE 0
```

This set of abstractions are used for processing incoming data and manipulating the PWM hardware interface. The following table describes each abstraction and its function.

Table 10 - Commands

Abstraction Name	Description
PWM	This is a command that tells the MQTT client the PWM state will not change binarily, but the duty cycle will change. This is accompanied by a value between 1 and 999 inclusive
TOGGLE	This is a command that tells the MQTT client that the PWM state will change binarily. This is accompanied by an ON or OFF value.
ON	A value indicating the PWM peripheral should be turned on
OFF	A value indicating the PWM peripheral should be turned off

Issued By:	Approved By:	Effective Date:	Page 38 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

Status	This is a command that tells the MQTT client to publish its current status to the broker. (not yet implemented)
--------	---

//MQTT CMD Type Abstractions

```
#define PWM '1'
#define TOGGLE '2'
#define ON '1'
#define OFF '0'
#define STATUS '3'
```

The following structure defines a new data-type required to process incoming data intended to manipulate the PWM peripheral. The following table describes each part and function.

Table 11 - Type MQTT_Message

Property Name	Data Type	Description
reserved	char[10]	The MQTT client isn't concerned with this data.
command	Char	Tells the client what type of packet it is looking at
data	Char[]	Contains a variable length char array which contains a command code and data supporting the command.

//Typedef for incoming MQTT data

```
typedef struct
```

```
{
    char reserved[10];
    char command;
    char data[];
}MQTT_Message;
```

```
/******
```

```
-----Function Prototypes-----
```

```
*****/
```

```
void Mqtt_Connect(SOCKET);
void Mqtt_Publish(SOCKET);
void Mqtt_Subscribe(SOCKET);
char Mqtt_ReadMessage(void*);
void Mqtt_Callback(SOCKET, void*);
```

5.1.2 MCQTT.c

MQTT.c contains the MQTT client. This client integrates with a hardware interface provided by the PWM driver and the socket callbacks provided by the WiFi stack.

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 39 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

5.1.2.1 MQTT STRINGS

The following variables are required for the MQTT client to function:

```
//MQTT strings
unsigned short packetID = 1;    //This is the current packetID for packets requiring one
char mqtt_Buffer[BUFFER_SIZE]; //This buffer is used for data to be sent
char mqtt_Broker_Address[] = "m12.cloudmqtt.com"; //This is to resolvable address of the
broker you want to connect to. Not #defined to allow programatic update
char mqtt_userName[] = "*****"; //This is the user ID associated with the module. Not
required for all implementations
char mqtt_Password[] = "*****"; //This is the password for said user ID.
char mqtt_ClientID[] = "edgetest"; //This is the Client ID. This may not be necessary
depending on your broker
char mqtt_Topic[] = "test/1"; //The topic you wish to publish to
char mqtt_Message[MAX_MESSAGE_SIZE] = "testing"; //The message you wish to send to
said topic. This is for testing only as the message should go directly into the buffer.
volatile int brightness;
```

5.1.2.1 FUNCTIONS

The Mqtt_Callback function is what makes the MQTT client event driven. This function takes in a void pointer to the socket buffer and determines what kind of data is in it. The buffer is masked and evaluated against known MQTT control packet types. Each defines case is a valid response from the broker

Notice that the socket buffer brought in as a void pointer is immediately case as tstrSocketRecvMsg pointer. This allows access to the payload without having to guess where it is in the buffer. Therefor, the payload can be masked and checked against known broker responses.

```
void Mqtt_Callback(SOCKET mqtt_sock, void* data)
{
    tstrSocketRecvMsg* msgData = (tstrSocketRecvMsg *)data;
    switch(*msgData->pu8Buffer & 0xF0)
    {
        case CONNACK_CB:
            //ToDo: Write function to handle all broker return codes for connack.
            //printf("MQTT: Connected to broker\r\n");

            //Subscribe to topic for this module
            Mqtt_Subscribe(mqtt_sock);
```

Issued By:	Approved By:	Effective Date:	Page 40 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

break;

case PUBACK_CB:
//ToDo: Write function to handle all broker return codes for puback
//printf("MQTT: Message Published \r\n");
break;

case SUBACK_CB:
//ToDo: Write function to handle all broker return codes for suback
//printf("MQTT: subscription accepted \r\n");
break;

case MESSAGE_RECEIVED:
//ToDo: Write function to handle all broker codes associated with messages
//printf("MQTT: Message received \r\n");
Mqtt_ReadMessage(msgData->pu8Buffer);
break;

default:
//ToDo: Write printf that presents the actual return code received. Develop against this
//printf("MQTT: Undefined response from broker. You should write this in");
break;
}
}

```

Mqtt_Connect is a long and complicated function. Instead of trying to explain the whole thing in one paragraph, it will be broken up into smaller chunks.

The Mqtt_Connect function builds a connection string and uses it to connect to the configured MQTT broker using socket information provided by the established ssl connection.

```
void Mqtt_Connect(SOCKET mqtt_sock)
```

First, the connect buffer is nulled out. This ensures no garbage data from the last transmission is accidentally sent out to the broker.

```

//make pointer and clear buffer
char* connectBuffer = (char*)&mqtt_Buffer;
memset(mqtt_Buffer, '0', BUFFER_SIZE);

```


Issued By:	Approved By:	Effective Date: 10/31/2018	Page 41 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

This block of code gets and calculates as many values as possible before building the connection string. This makes the rest of this code much easier to read and more resource efficient from an execution standpoint as these values only need to be calculated one time.

```
//get string lengths
unsigned short intBuffer;
char* ptrIntBuffer = (char*)&intBuffer;
unsigned short passLength = strlen(mqtt_Password);
unsigned short userLength = strlen(mqtt_userName);
unsigned short protocolNameLength = strlen(PROTOCOL_NAME);
unsigned short clientIdLength = strlen(mqtt_ClientID);
```

This line compiles the fixed header for the connection string. Notice the abstractions described earlier are in use, making this very easy to read and understand. As the buffer is populated, the pointer value is stepped up using known values and hard coded values.

```
/Packet type and fixed flags
*connectBuffer = (PACKET_TYPE(CONNECT) | FH_DUP(FALSE) |
FH_QOS(ONCE_AT_MOST) | FH_RETAIN(FALSE));
```

This line calculates the remaining length for the packet. This is required for the broker to know where data begins and ends. The values used to calculate this value were calculated in the second block of code looked at.

```
//Remaining Length
*(connectBuffer + 1) = passLength + userLength + protocolNameLength + clientIdLength +
12;
```

This block stores the protocol name length into the connect buffer. Because of a difference in endianness, the integer needs to have its high byte swapped with the low byte. The protocol length is stored in a temporary container, and has its high and low bytes swapped. Then those bytes are added to the connect buffer one at a time. In hind sight, this could have been done without swapping those bytes first.

```
//Protocol name length
intBuffer = protocolNameLength;
intBuffer = ((intBuffer<<8)&0xff00)|((intBuffer>>8)&0x00ff);
*(connectBuffer + 2) = *ptrIntBuffer;
*(connectBuffer + 3) = *(ptrIntBuffer + 1);
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 42 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

//Protocol name

strcpy(connectBuffer + 4, PROTOCOL_NAME);

//protocol level

*(connectBuffer + 4 + protocolNameLength) = PROTOCOL_LEVEL;

This section builds the Variable Header. Again, the use of abstraction continues throughout the process. There are very few hard-coded values.

//Variable Header flags

*(connectBuffer + 5 + protocolNameLength) = (VH_USER_NAME(TRUE) |
VH_PASS(TRUE) | VH_QOS(ONCE_AT_MOST) | VH_WILL_RETAIN(FALSE) |
VH_WILL_FLAG(FALSE) | VH_CLEAN_SESSION(TRUE));

//keep alive

intBuffer = KEEP_ALIVE;

intBuffer = ((intBuffer<<8)&0xff00)|((intBuffer>>8)&0x00ff);

*(connectBuffer + 6 + protocolNameLength) = *ptrIntBuffer;

*(connectBuffer + 7 + protocolNameLength) = *(ptrIntBuffer + 1);

//ClientID length

intBuffer = clientIdLength;

intBuffer = ((intBuffer<<8)&0xff00)|((intBuffer>>8)&0x00ff);

*(connectBuffer + 8 + protocolNameLength) = *ptrIntBuffer;

*(connectBuffer + 9 + protocolNameLength) = *(ptrIntBuffer + 1);

When adding a string to the buffer, string copy is the best way to do it. String copy automates the byte-by-byte copy for a char array into a buffer.

//clientId

strcpy(connectBuffer + 10 + protocolNameLength, mqtt_ClientID);

//User ID Length

intBuffer = userLength;

intBuffer = ((intBuffer<<8)&0xff00)|((intBuffer>>8)&0x00ff);

*(connectBuffer + 10 + protocolNameLength + clientIdLength) = *ptrIntBuffer;

*(connectBuffer + 11 + protocolNameLength + clientIdLength) = *(ptrIntBuffer + 1);

//User ID

strcpy(connectBuffer + 12 + clientIdLength + protocolNameLength, mqtt_userName);

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 43 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
//Password Length
intBuffer = passLength;
intBuffer = ((intBuffer<<8)&0xff00)|((intBuffer>>8)&0x00ff);
*(connectBuffer + 12 + clientIdLength + userLength + protocolNameLength) =
*ptrIntBuffer;
*(connectBuffer + 13 + clientIdLength + userLength + protocolNameLength) =
*(ptrIntBuffer + 1);

//Password
strcpy(connectBuffer + 14 + clientIdLength + userLength + protocolNameLength,
mqtt_Password);
```

This is the last block in the function and sends the completed buffer to the MQTT broker then increments the packetID. Each packet sent to the broker on a single socket must have a unique packet ID.

```
//send connect packet to broker using current web socket
send(mqtt_sock, connectBuffer, (*(connectBuffer + 1) + 2),0);
packetID++;
```

The Mqtt Publish function publishes data to the broker. Building the fixed and Variable Headers are the same in practice, but different values are used. The same is true for the payload. Notice the progression of this function is very close to the progression of the last function.

```
void Mqtt_Publish(SOCKET mqtt_sock)
```

```
//Some working variables
unsigned short messageLength = strlen(mqtt_Message);
unsigned short topicLength = strlen(mqtt_Topic);
unsigned short intBuffer;

//Some working pointers. This is required to index one byte at a time for short data-types
char* sendbuffer = (char*)&mqtt_Buffer; //point to buffer
char* ptrPID = (char*)&packetID; //point to PID
char* ptrIntBuffer = (char*)&intBuffer;

//Clear buffer PUT 1024 in #define
memset(mqtt_Buffer, '\0', BUFFER_SIZE); //clear buffer with '\0's

//Build fixed header
*sendbuffer = (PACKET_TYPE(PUBLISH) | FH_DUP(FALSE) |
FH_QOS(ONCE_AT_LEAST) | FH_RETAIN(TRUE));
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 44 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

*(sendbuffer + 1) = (topicLength + messageLength + 4);

//build Variable Header (topic and packed ID)
intBuffer = topicLength;
intBuffer = ((intBuffer<<8)&0xff00)|((intBuffer>>8)&0x00ff);
*(sendbuffer + 2) = *(ptrIntBuffer);
*(sendbuffer + 3) = *(ptrIntBuffer + 1);
strcpy(sendbuffer + 4, mqtt_Topic);

//Assign packet identifier
packetID = ((packetID<<8)&0xff00)|((packetID>>8)&0x00ff);
*(sendbuffer + 4 + topicLength) = *(ptrPID);
*(sendbuffer + 5 + topicLength) = *(ptrPID + 1);

//Build payload (message)
strcpy(sendbuffer + 6 + topicLength, mqtt_Message);

//debug
//printf("MQTT: Publishing Message: %s\r\n", mqtt_Message);

//Publish message to specified topic on specified broker
send(mqtt_sock, sendbuffer, (*(sendbuffer + 1) + 2),0);
packetID++;

```

The Mqtt_Subscribe function tells the broker this module wants to subscribe to a topic. Again, this function follows a very familiar pattern.

```

void Mqtt_Subscribe(SOCKET mqtt_sock)

//Clear buffer
memset(mqtt_Buffer, '0', BUFFER_SIZE);

//working variables
unsigned short intBuffer;
unsigned short topicLength = strlen(mqtt_Topic);

//Some working pointers. This is required to index one byte at a time for short data-types
char* ptrIntBuffer = (char*)&intBuffer;
char* subscribeBuffer = (char*)&mqtt_Buffer;

//Fixed header type and flags. for MQTT version 3.1.1 this must be 0x82

```

Issued By:	Approved By:	Effective Date:	Page 45 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
*subscribeBuffer = (PACKET_TYPE(SUBSCRIBE) | FH_DUP(0) |
FH_QOS(ONCE_AT_LEAST) | FH_RETAIN(FALSE));
```

```
//Remaining length
```

```
*(subscribeBuffer + 1) = topicLength + 5;
```

```
//Assign packet identifier
```

```
intBuffer = ((packetID<<8)&0xff00)|((packetID>>8)&0x00ff);
```

```
*(subscribeBuffer + 2) = *(ptrIntBuffer);
```

```
*(subscribeBuffer + 3) = *(ptrIntBuffer + 1);
```

```
//Topic filter Length
```

```
intBuffer = topicLength;
```

```
intBuffer = ((topicLength<<8)&0xff00)|((topicLength>>8)&0x00ff);
```

```
*(subscribeBuffer + 4) = *(ptrIntBuffer);
```

```
*(subscribeBuffer + 5) = *(ptrIntBuffer + 1);
```

```
//Topic name
```

```
strcpy(subscribeBuffer + 6, (char*)mqtt_Topic);
```

```
//Requested QoS
```

```
*(subscribeBuffer + 6 + topicLength) = ONCE_EXACTLY;
```

```
send(mqtt_sock, subscribeBuffer, *(subscribeBuffer + 1) + 2, 0);
```

The Mqtt_ReadMessage function departs from the form seen in the last three functions. The purpose of this function is to take a command message, parse it, and call the appropriate PWM hardware interface function to manipulate the PWM peripheral.

```
char Mqtt_ReadMessage(void* msg)
```

First, the void pointer is cast as an MQTT message so it can be parsed.

```
MQTT_Message* msgData = (MQTT_Message *)msg;
```

```
//printf("MQTT: Parsing Message \r\n");
```

The switch case checks what command was sent using the structure given to the previously void data.

```
switch(msgData->command)
{
```

```
    //Case for change to brightness of LED
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 46 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

case PWM:
    //Debug info
    //printf("EdgeModule: PWM Command Received %c%c%c \r\n", msgData->data[0],
msgData->data[1], msgData->data[2]);

```

```

    //Add null terminator for conversion to unsigned long
    msgData->data[3] = 0;

```

Since the data comes in as a char array, it must be converted to an unsigned long to fill size of the array.

```

    //Store brightness for later
    brightness = strtol(msgData->data, 0l, 10);

```

```

    //set the duty-cycle
    SetBrightness(brightness);
    break;

```

```

    //Case for boolean change (ON/OFF) to LED

```

```

case TOGGLE:
    switch(msgData->data[0])
    {
        //Enables the PWM peripheral
        case ON:
            //printf("EdgeModule: PWM Command Received (ON) \r\n");
            LEDOn(brightness);
            break;

            //Disables the PWM peripheral
        case OFF:
            //printf("EdgeModule: PWM Command Received (OFF) \r\n");
            LEDOff();
            break;

        default:
            break;
    }
    break;

```

```

case STATUS:
    //ToDo respond with status of PWM
    break;

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 47 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

//default to catch unexpected command codes. Used for error handling
default:
    //printf("EdgeModule: Command not processed " + msgData->data[0]);
    break;
}
//extract the message and store it in the buffer
//ToDo: parse message and take action if needed
return 1 ;

```

5.1.3 PWM.h

This file contains only function prototypes. These are required for the compiler to know what functions look like when scanning code in PWM.c

```

void StartPWM(void);
void SetBrightness(unsigned int);
void LEDOff(void);
void LEDOn(unsigned int);

```

5.1.4 PWM.c

The StartPWM function initializes the PWM peripheral device and attaches the desired port-pin to it. This function does not actually start the PWM peripheral because the desired default state is off.

Note the use in Group[]. This is how the correct multiplexed pin is targeted. First the pin group is selected, then the port-pin in that group is selected. The line...

```

PORT->Group[1].PMUX[6].bit.PMUXE = PORT_PMUX_PMUXE_F; //Set peripheral to
multiplexer output pin

```

...Illustrates this well.

```

void StartPWM()
{
    //Port Pin configuration
    PORT->Group[1].DIRSET.reg = PORT_PB12; //Set pin direction
    PORT->Group[1].OUTCLR.reg = PORT_PB12; //Set up control register
    PORT->Group[1].PINCFG[12].reg |= PORT_PINC_CFG_PMUXEN; //Attach pin to
multiplexer
    PORT->Group[1].PMUX[6].bit.PMUXE = PORT_PMUX_PMUXE_F; //Set peripheral to
multiplexer output pin

```

Issued By:	Approved By:	Effective Date:	Page 48 of
		10/31/2018	91
Travis Cucore	Travis Cucore	Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

```

// power on the device
PM->APBCMASK.reg |= PM_APBCMASK_TCC0;
GCLK->CLKCTRL.reg = (GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK0
| GCLK_CLKCTRL_ID(TCC0_GCLK_ID));
while (GCLK->STATUS.bit.SYNCBUSY) {} //Wait for synchronization

// reset TCC module
TCC0->CTRLA.reg = TCC_CTRLA_SWRST; //Force TCC Reset
while (TCC0->SYNCBUSY.reg & TCC_SYNCBUSY_SWRST) {} //Check for sync
TCC0->CTRLBCLR.reg = TCC_CTRLBCLR_DIR; // Tell TCC0 to count up
while (TCC0->SYNCBUSY.reg & TCC_SYNCBUSY_CTRLB) {} //Wait for
synchronization

//configure the TCC device
TCC0->CTRLA.reg = (TCC_CTRLA_PRESCSYNC_GCLK_Val |
TCC_CTRLA_PRESCALER(TCC_CTRLA_PRESCALER_DIV1_Val)); //Set prescale and
general clock frequency

// select the waveform generation mode -> normal PWM
TCC0->WAVE.reg = (TCC_WAVE_WAVEGEN_NPWM); //Tell peripheral device to
operate in NPWM mode
while (TCC0->SYNCBUSY.reg & TCC_SYNCBUSY_WAVE) {} //Wait for
synchronization

// set the selected period
TCC0->PER.reg = (999); //Set period of timer (999 ticks)
while (TCC0->SYNCBUSY.reg & TCC_SYNCBUSY_PER) {} //Wait for synchronization
}

```

SetBrightness sets the desired duty-cycle in the counter compare buffer. The value set is propagated to the active counter compare register on the next synchronization. Writing directly to the counter compare register can halt the timer.

```

void SetBrightness(unsigned int brightness)
{
    TCC0->CCB[2].reg = brightness; //Set duty-cycle
}

```

LEDOff disables the PWM peripheral by clearing the timer counter enable bit in the control register.

Issued By:	Approved By:	Effective Date:	Page 49 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
void LEDOff()
{
    TCC0->CTRLA.reg &= ~(0x1ul << TCC_CTRLA_ENABLE_Pos); //Disable PWM peripheral
}
```

LEDOOn first sets the counter compare buffer with the current brightness value, then enables the PWM peripheral by setting the PWM peripheral enable bit in the timer counter control register.

```
void LEDOn(unsigned int brightness)
{
    TCC0->CCB[2].reg = brightness; //Set brightness in CC buffer to avoid sync issues
    TCC0->CTRLA.reg |= (TCC_CTRLA_ENABLE); //Enable PWM peripheral
}
```

5.1.5 main.h

This file contains some critical abstractions, but in order to discuss it legally, the license needs to be presented with it.

```
/**
 * \file
 *
 * \brief MAIN configuration.
 *
 * Copyright (c) 2016 Atmel Corporation. All rights reserved.
 *
 * \asf_license_start
 *
 * \page License
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 *
```

Issued By:	Approved By:	Effective Date:	Page 50 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.: 0001	Version: 01

Internet of Things Proof of Concept

* 3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.

* THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

* \asf_license_stop

*/

Some erroneous computer-generated code has been removed to shorten this section. The following #defines determine how a connection is to be established to an AP as well as the target SSL remote host.

```

/** Wi-Fi Settings */
#define MAIN_WLAN_SSID "SpareRoom-2.4Ghz" /**< Destination SSID */
#define MAIN_WLAN_SSID "Cucore_AdHoc" /**< Destination SSID */
#define MAIN_WLAN_AUTH M2M_WIFI_SEC_WPA_PSK /**< Security manner */
#define MAIN_WLAN_PSK "Cmj893e!" /**< Password for Destination SSID */

/** Using IP address. */
#define IPV4_BYTE(val, index) ((val >> (index * 8)) & 0xFF)

```

Issued By:	Approved By:	Effective Date:	Page 51 of 91
Travis Cuore	Travis Cuore	10/31/2018	
		Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
/** All SSL defines */
#define MAIN_HOST_NAME      "m12.cloudmqtt.com"
#define MAIN_HOST_PORT      24500
```

These enumerations are used to track the state of any given active socket.

```
typedef enum {
    SocketInit = 0,
    SocketConnect,
    SocketWaiting,
    SocketComplete,
    SocketError
} eSocketStatus;
```

5.1.6 main21.c

This section will be broken up into several parts due to its size, but first the Atmel license must be presented.

```
\file
*
* \brief SSL Example.
*
* Copyright (c) 2016 Atmel Corporation. All rights reserved.
*
* \asf_license_start
*
* \page License
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
* 1. Redistributions of source code must retain the above copyright notice,
*    this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright notice,
*    this list of conditions and the following disclaimer in the documentation
*    and/or other materials provided with the distribution.
*
* 3. The name of Atmel may not be used to endorse or promote products derived
*    from this software without specific prior written permission.
*
```

Issued By:	Approved By:	Effective Date:	Page 52 of 91
Travis Cucore	Travis Cucore	10/31/2018	
		Document No.:	Version:
		0001	01

Internet of Things Proof of Concept

* THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE
 * EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
 * ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *
 * \asf_license_stop

Some computer generated, and code used for compiling have been skipped to avoided unnecessary discussion.

Socket_cb is a function the MQTT client hooks into. This function provides event handling for SSL sockets. Notice the socket buffer is brought in as a void pointer. This makes it extremely easy to cast unknown data structures differently as they are passed around the application. Also notice the SOCKET data-type. This contains all data required to describe the state of a socket and connection information. This is also used by the MQTT client for sending data to the broker over an established socket.

When the socket_cb executes the socket message connect state, the MQTT broker uses the provided socket information to establish a connection to the broker. At this point the device is connected to the broker server, but not to the broker.

When the socket indicates the WINC1500 module has a new message pending. The recv() function is called to transfer data to the MCU for processing. Once this is done, the MQTT client passes the void pointer to the MQTT callback function to begin processing.

```
static void socket_cb(SOCKET sock, uint8_t u8Msg, void *pvMsg)
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 53 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

{
    /* Check for socket event on TCP socket. */
    if (sock == tcp_client_socket)
    {
        switch (u8Msg)
        {
            case SOCKET_MSG_CONNECT:
            {
                tstrSocketConnectMsg *pstrConnect = (tstrSocketConnectMsg *)pvMsg;
                if (pstrConnect && pstrConnect->s8Error >= 0)
                {
                    printf("socket_cb: connect success!\r\n");

                    //Anytime the client loses connection it will attempt to connect to the broker
                    when connectivity is restored.
                    //The new connection to the broker is always a clean connection as described
                    in the MQTT 3.1.1 specification.
                    Mqtt_Connect(sock); //connect to broker

                    recv(sock, pvMsg, BUFFER_SIZE, RECV_TIMEOUT); //wait for ack
                }
                else
                {
                    printf("socket_cb: connect error!\r\n");
                    close(tcp_client_socket);
                    tcp_client_socket = -1;
                }
            }
        }
        break;

        /* Message send */
        case SOCKET_MSG_SEND:
        {
            printf("socket_cb: send success!\r\n");
        }
        break;

        /* Message receive */
        case SOCKET_MSG_RECV:
        {
            tstrSocketRecvMsg *pstrRecv = (tstrSocketRecvMsg *)pvMsg;

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 54 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

if (pstrRecv && pstrRecv->s16BufferSize > 0)
{
    printf("socket_cb: recv success!\r\n");
    recv(sock, pstrRecv->pu8Buffer, BUFFER_SIZE, RECV_TIMEOUT);
    Mqtt_Callback(sock, pstrRecv);
}
else
{
    printf("socket_cb: recv error!\r\n");
    close(sock);
}
}
}

break;

case SOCKET_MSG_RECVFROM:
{
    ;
}
break;
default:
{
    break;
}
}
}
}

```

The `wifi_cb` function is called when a change in connection state or the module needs to go through the DHCP process. In short, this function provides hooks for application layer code to perform actions based on the state of the WiFi module. This project does not use these hooks but could in the future to make the system more robust.

```

static void wifi_cb(uint8_t u8MsgType, void *pvMsg)
{
    switch (u8MsgType) {
        case M2M_WIFI_RESP_CON_STATE_CHANGED:
        {
            tstrM2mWifiStateChanged *pstrWifiState = (tstrM2mWifiStateChanged *)pvMsg;
            if (pstrWifiState->u8CurrState == M2M_WIFI_CONNECTED) {

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 55 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

        printf("wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED:
CONNECTED\r\n");
        m2m_wifi_request_dhcp_client();
    } else if (pstrWifiState->u8CurrState == M2M_WIFI_DISCONNECTED) {
        printf("wifi_cb: M2M_WIFI_RESP_CON_STATE_CHANGED:
DISCONNECTED\r\n");
        gbConnectedWifi = false;
        gbHostIpByName = false;
        m2m_wifi_connect((char *)MAIN_WLAN_SSID, sizeof(MAIN_WLAN_SSID),
MAIN_WLAN_AUTH, (char *)MAIN_WLAN_PSK,
M2M_WIFI_CH_ALL);
    }

    break;
}

case M2M_WIFI_REQ_DHCP_CONF:
{
    uint8_t *pu8IPAddress = (uint8_t *)pvMsg;
    /* Turn LED0 on to declare that IP address received. */
    printf("wifi_cb: M2M_WIFI_REQ_DHCP_CONF: IP is %u.%u.%u.%u\r\n",
pu8IPAddress[0], pu8IPAddress[1], pu8IPAddress[2], pu8IPAddress[3]);
    gbConnectedWifi = true;

    /* Obtain the IP Address by network name */
    gethostbyname((uint8_t *)MAIN_HOST_NAME);
    break;
}

default:
{
    break;
}
}
}

```

Everything else takes place in the main program function. This function calls all of the initialization subroutines and starts the main program loop which looks for a socket error and tries to reinitialize the socket if an error exists.

```
int main(void)
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 56 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

{
    tstrWifiInitParam param;
    int8_t ret;

    /* Initialize the board. */
    system_init();

    /* Initialize the UART console. */
    configure_console();
    printf(STRING_HEADER);

    /* Initialize the BSP. */
    nm_bsp_init();

    /* Initialize Wi-Fi parameters structure. */
    memset((uint8_t *)&param, 0, sizeof(tstrWifiInitParam));

    /* Initialize Wi-Fi driver with data and status callbacks. */
    param.pfAppWifiCb = wifi_cb;
    ret = m2m_wifi_init(&param);
    if (M2M_SUCCESS != ret) {
        printf("main: m2m_wifi_init call error!(%d)\r\n", ret);
        while (1) {
        }
    }

    /* Initialize Socket module */
    socketInit();
    registerSocketCallback(socket_cb, resolve_cb);

    /* Connect to router. */
    m2m_wifi_connect((char *)MAIN_WLAN_SSID, sizeof(MAIN_WLAN_SSID),
        MAIN_WLAN_AUTH, (char *)MAIN_WLAN_PSK, M2M_WIFI_CH_ALL);

    //Start PWM Peripheral
    StartPWM();

    //Main program loop
    while (1) {
        m2m_wifi_handle_events(NULL);
    }
}

```


Issued By:	Approved By:	Effective Date: 10/31/2018	Page 57 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

if (gbConnectedWifi && gbHostIpByName) {
    if (gu8SocketStatus == SocketInit) {
        if (tcp_client_socket < 0) {
            gu8SocketStatus = SocketWaiting;
            if (sslConnect() != SOCK_ERR_NO_ERROR) {
                gu8SocketStatus = SocketInit;
            }
        }
    }
}
}
}
}

return 0;
}

```

5.2 Android Application

This Android application was built on the Xamarin Forms and .NET frameworks using the Visual Studio IDE. This section will only discuss those files written for this project as there are far too many dependencies to reasonably discuss.

5.2.1 Views

On the Android platform, a view is a page a user interacts with. Each view consists of a UI written in xaml and code-behind written in C#. Code-behind works like glue between the user experience and application processes.

5.2.1.1 AddModule.xaml

This view is presented to the user when they choose to add a new module or edit an existing module.

```

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="EdgeNet.Views.AddModule">
    <ContentPage.ToolbarItems>

```

The toolbar item defaults to “Delete Module” text, but is changed in code-behind depending on context.

```

        <ToolbarItem x:Name="deleteModuleTBItem" Order="Primary" Text="Delete Module"
        Activated="DeleteModuleTBItem_Activated"/>
    </ContentPage.ToolbarItems>

```

A stack layout is a stackable framework wherein all objects are automatically stacked on top of each other. This stack layout is padded 30 pixels from each edge of the screen and will fill and expand to fit that space.

Issued By:	Approved By:	Effective Date:	Page 58 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
<StackLayout Padding="30" HorizontalOptions="FillAndExpand"
VerticalOptions="FillAndExpand">
```

An entry is an input allowing text to be collected from the user. This entry is bound to the Name property of the Module passed to the view when it was created. As data is entered into the textbox it is propagated in real-time back to the name property of the Module object.

```
<Entry Placeholder="Device Name" x:Name="moduleNameEntry" Text="{Binding
Name}" PlaceholderColor="SlateGray" TextColor="DarkSlateGray"/>
```

A picker is a UI element that allows the user to pick an item populated as a choice. It defaults to index of 0 and can be dynamically populated. In this case it will populate with known locations and the option to create a new location. This object is bound to the location property of the Module passed to the view when it was created.

```
<Picker x:Name="moduleLocationPicker" TextColor="SlateGray" IsEnabled="True"
Title="Select Location" SelectedItem="{Binding Location}"
SelectedIndexChanged="OnLocationSelected"/>
```

This entry is populated by the above picker and is not visible until a selection is made. This object is also bound to the location property of the Module object passed to the view when it was created.

```
<Entry Placeholder="Device Location" IsEnabled="False" IsVisible="False"
x:Name="moduleLocationEntry" Text="{Binding Location}" PlaceholderColor="SlateGray"
TextColor="DarkSlateGray"/>
```

This picker operates the same as the last picker except it is bound to the room property of the Module object passed to the view when it was created.

```
<Picker x:Name="moduleRoomPicker" TextColor="SlateGray" IsEnabled="True"
Title="Select Room" SelectedItem="{Binding Room}"
SelectedIndexChanged="OnRoomSelection"/>
```

This entry works like all other entries. Note the placeholder that shows up before any data is entered by the user.

```
<Entry Placeholder="Room" x:Name="moduleRoomEntry" IsVisible="False"
IsEnabled="False" Text="{Binding Room}" PlaceholderColor="SlateGray"
TextColor="DarkSlateGray"/>
<Entry Placeholder="Device ID" x:Name="moduleIDEntry" Text="{Binding MID}"
PlaceholderColor="SlateGray" TextColor="DarkSlateGray"/>
<Picker x:Name="identityPicker" IsEnabled="True" Title="Select Module Identity"
SelectedItem="{Binding Type}" TextColor="SlateGray"/>
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 59 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

The button object is associated with an event handler called “DoneBtn_Clicked” in code-behind. This link must be made here in xaml before it can be used. The text property sets what text is seen on the button.

```
<Button Text="Done" Clicked="DoneBtn_Clicked"/>
```

This box view and accompanying image contain the EdgeNet logo. Every view has one of these pairs.

```
<BoxView VerticalOptions="FillAndExpand"/>
<Image Source="Logo.png" HorizontalOptions="FillAndExpand"
VerticalOptions="End"/>
</StackLayout>
</ContentPage>
```

5.2.1.2 AddModule.xaml.cs

This file contains code-behind for the AddModule user interface defined in the xaml file. It provides linkage to the rest of the application. The methods and other contents of this view are broken out of the namespace container for ease of viewing in this document. These code-behind files are pretty well commented, so only comments adding value to the document will be added.

//Hard coded list of supported module types. This is used to populate the module type dropdown box

```
List<string> typePickerList = new List<string>()
{
    "Light",
    "Dimming Light",
    "Ceiling Fan",
    "Thermostat"
};
```

```
//This is passed in from the last activity
bool isModuleNew;
```

```
//Instantiate instance of Module to for data bindings
volatile static Module module;
```

```
public AddModule (bool isNew, Module thisModule)
{
    InitializeComponent();
```

```
//Passed in from last activity. Will be null if module is new.
module = thisModule;
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 60 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

This line links the list created at the start of this class with the module type picker.

```
identityPicker.ItemsSource = typePickerList;
//Set the title of the navigation page
this.Title = "Add Module";

//This is passed in from the last activity
this.isModuleNew = isNew;

//find out if the module is new to the system
if (isModuleNew)
{
    ToolbarItems.Remove(deleteModuleTBItem);
}
}

//Tasks to perform when the activity appears on screen
protected async override void OnAppearing()
{
    base.OnAppearing();

    //Bring in all of the databases in order to populate the form and to enable saving the new
    or revised module.
    List<Module> modulesList = await App.modDatabase.GetModulesAsync();
    List<string> locations = modulesList.Select(x => x.Location).Distinct().ToList<string>();
    locations.Add("New Location"); //Just in case the desired location does not exist yet
    List<string> rooms = modulesList.Select(x => x.Room).Distinct().ToList<string>();
    rooms.Add("New Room"); //Just in case the desired room does not exist yet.

    //Find out if the module is new
    if(!isModuleNew)
    {
        //get index of current room in pickerlist
        int roomIndex = -1;
        foreach (string element in rooms)
        {
            roomIndex++;

            if (element == module.Room)
            {
                break;
            }
        }
    }
}
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 61 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

    }
}

//get index of current location in pickerlist
int locationIndex = -1;
foreach (string element in rooms)
{
    locationIndex++;

    if (element == module.Location)
    {
        break;
    }
}

//Bind lists to pickers
moduleLocationPicker.ItemsSource = locations;
moduleRoomPicker.ItemsSource = rooms;

//Set default index so they always populate i
moduleLocationPicker.SelectedIndex = locationIndex;
moduleRoomPicker.SelectedIndex = roomIndex;
}
else
{
    //Bind lists to pickers
    moduleLocationPicker.ItemsSource = locations;
    moduleRoomPicker.ItemsSource = rooms;
}
}

//Event handler for delete module toolbar button
async void DeleteModuleTBIItem_Activated(object sender, EventArgs e)
{
    var newModule = (Module)BindingContext; //store data bindings for use
    await App.modDatabase.DeleteModuleAsync(newModule); //save data to database
    await Navigation.PopToRootAsync(); //navigate back to the screen you came from
}

//Event handler for done button click event

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 62 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

async void DoneBtn_Clicked(object sender, EventArgs e)
{
    var newModule = (Module)BindingContext; //store data bindings for use
    await App.modDatabase.SaveModuleAsync(newModule); //save data to database
    moduleLocationEntry.IsVisible = false;
    moduleLocationEntry.IsEnabled = false;
    moduleLocationPicker.IsVisible = true;
    moduleLocationPicker.IsEnabled = true;
    moduleLocationEntry.IsVisible = false;
    moduleLocationEntry.IsEnabled = false;
    moduleLocationPicker.IsVisible = true;
    moduleLocationPicker.IsEnabled = true;
    await Navigation.PopAsync(); //navigate back to the screen you came from
}

//Event handler for the location picker
void OnLocationSelected(object sender, SelectedItemChangedEventArgs e)
{
    //Check if the user has selected a new location. If so, change focus, disable and hide
    pickers while enabling and making the entry visible
    if(moduleLocationPicker.SelectedItem.Equals("New Location"))
    {
        moduleLocationEntry.IsVisible = true;
        moduleLocationEntry.IsEnabled = true;
        moduleLocationPicker.IsVisible = false;
        moduleLocationPicker.IsEnabled = false;
        moduleLocationEntry.Focus();
    }
}

//Event handler for the room picker
void OnRoomSelection(object sender, SelectedItemChangedEventArgs e)
{
    //Check if the user has selected a new room. If so, change focus, disable and hide picker
    while enabling and making the entry visible.
    if(moduleRoomPicker.SelectedItem.Equals("New Room"))
    {
        moduleRoomEntry.IsVisible = true;
        moduleRoomEntry.IsEnabled = true;
        moduleRoomPicker.IsVisible = false;
        moduleRoomPicker.IsEnabled = false;
    }
}

```

Issued By:	Approved By:	Effective Date:	Page 63 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

        moduleRoomEntry.Focus();
    }
}

```

5.2.1.3 EdgeNetMainPage.xaml

This view is a tabbed page. Each tab has a separate section of the xaml file. Elements of this tabbed view repeated across tabs will only be discussed in the first occurrence.

```

<?xml version="1.0" encoding="utf-8"?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="EdgeNet.Views.Testing.EdgeNetMainPage"
    Title="EdgeNet"
    x:Name="EdgeNetTabbedView"
    Padding="15">

```

This list indicates that the tabbed page has children and their definitions are about to start.

```

<TabbedPage.Children>

```

Skipping this section because it is not implemented.

```

    <ContentPage Title="Profiles" x:Name="profilesTab">
        <ContentPage.ToolbarItems>
            <ToolbarItem
                Order="Primary"
                Text="Add/Edit"
                Activated="OnAddRemoveProfile"/>
        </ContentPage.ToolbarItems>
        <StackLayout>
            <Label x:Name="currentProfileLabel" Text="Current Profile" VerticalOptions="Start"
HorizontalOptions="CenterAndExpand" FontSize="36" TextColor="#00A79D"/>
            <ListView x:Name="ProfileListView"
                HorizontalOptions="FillAndExpand"
                VerticalOptions="FillAndExpand"
                RowHeight="80">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <TextCell Text="{Binding .}" TextColor="#00A79D"/>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
            <Image Source="Logo.png" HorizontalOptions="FillAndExpand"
VerticalOptions="End"/>

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 64 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
</StackLayout>
</ContentPage>
```

This section defines the locations tab. It consists of a stack layout with a list view in it. A list view is a scrollable list populated with data users can select from.

```
<ContentPage Title="Locations" x:Name="locationsTab">
  <StackLayout>
    <ListView x:Name="LocationListView"
      HorizontalOptions="FillAndExpand"
      VerticalOptions="FillAndExpand"
      ItemSelected="LocationItemSelected"
      RowHeight="80">
```

The list view item template determines what data will be displayed in what manner on the list. Notice the binding does not call out anything specific. That is because in this case, the binding is called out in code-behind, because at the time this view is created, data may not be available yet.

```
    <ListView.ItemTemplate>
      <DataTemplate>
        <TextCell Text="{Binding .}" TextColor="#00A79D"/>
      </DataTemplate>
    </ListView.ItemTemplate>
  </ListView>
```

At the bottom of every list view is a save profile button that does nothing right now and the EdgeNet logo.

```
    <Button x:Name="saveGlobalProfileButton" Text="Save Global Profile"/>
    <Image Source="Logo.png" HorizontalOptions="FillAndExpand"
      VerticalOptions="End"/>
  </StackLayout>
</ContentPage>
```

This section starts the rooms tab layout.

```
<ContentPage Title="Rooms" x:Name="roomsTab">
  <ContentPage.ToolbarItems>
```

The toolbar is changed to show an add module button because in this context it is appropriate. Its event handler is call OnAddModule_Activated and is tied to the activated property of the button.

```
    <ToolbarItem x:Name="addModule"
      Order="Primary"
      Text="Add Module"
```


Issued By:	Approved By:	Effective Date: 10/31/2018	Page 65 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

        Activated="OnAddModule_Activated"/>
    </ContentPage.ToolbarItems>
    <StackLayout>
        <ListView x:Name="ModuleListView"
            HorizontalOptions="FillAndExpand"
            VerticalOptions="FillAndExpand"
            ItemSelected="ItemSelected"
            RowHeight="80">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell Text="{Binding .}" TextColor="#00A79D"/>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
        <Button x:Name="saveLocationProfileButton" Text="Save Location Profile"/>
        <Image Source="Logo.png" HorizontalOptions="FillAndExpand"
VerticalOptions="End"/>
    </StackLayout>
</ContentPage>

```

The settings tab starts here. This tab is more like the add module view discussed before but there are some differences. In this case, two of the entry fields are marked as password fields. This tells the view to populate asterisks instead of what the user types in.

```

<ContentPage Title="Settings" x:Name="settingsTab">
    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="editSettingsBtn"
            Order="Primary"
            Text="Edit Settings"
            Activated="OnEditSettings_Activated"/>
    </ContentPage.ToolbarItems>
    <StackLayout>
        <Entry x:Name="brokerAddressEntry" IsEnabled="False" Text="{Binding
BrokerAddress}" Placeholder="Broker Address" PlaceholderColor="#00A79D"
TextColor="DarkSlateGray"/>
        <Entry x:Name="brokerPort" IsEnabled="False" Text="{Binding BrokerPort}"
Placeholder="Port Number" TextColor="DarkSlateGray" PlaceholderColor="#00A79D"/>
        <Entry x:Name="brokerUserNameEntry" IsEnabled="False" Text="{Binding
BrokerUserID}" Placeholder="Broker Username" PlaceholderColor="#00A79D"
TextColor="DarkSlateGray"/>
    </StackLayout>
</ContentPage>

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 66 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

    <Entry x:Name="brokerPasswordEntry" IsEnabled="False" Text="{Binding
BrokerPassword}" Placeholder="Broker Password" IsPassword="True"
PlaceholderColor="#00A79D" TextColor="DarkSlateGray"/>
    <Entry x:Name="brokerPasswordEntryTwo" IsEnabled="False" Text="{Binding
BrokerPassword}" Placeholder="Confirm Password" IsPassword="True"
PlaceholderColor="#00A79D" TextColor="DarkSlateGray"/>
    <Picker x:Name="defaultLocationPicker" IsEnabled="false" Title="Choose Default
Location" SelectedItem="{Binding DefaultLocation}" TextColor="SlateGray" />
    <Button x:Name="saveSettingsButton" Text="Save Settings"
Clicked="SaveBtn_Clicked" IsEnabled="False"/>
    <BoxView VerticalOptions="FillAndExpand"/>
    <Image Source="Logo.png" HorizontalOptions="FillAndExpand"
VerticalOptions="End"/>
</StackLayout>
</ContentPage>
</TabbedPage.Children>
</TabbedPage>

```

5.2.1.4 EdgeNetMainPage.xaml.cs

The edge net main page view is the main landing page for the EdgeNet application. For this reason, there is a lot going on this page unrelated to the user interface. Again, clearly commented code will not be restated. Only comments adding value to the document will be added.

//Instantiate objects to represent app settings and available modules.

```

public AppSettings currentSettings = new AppSettings();
public List<Module> modules = new List<Module>();

public EdgeNetMainPage ()
{
    InitializeComponent();
}

```

This is executed every time the view appears on the screen. This includes any time the user navigates away to add a module or look at the module list. This does not include switching from on tab to another.

```

protected async override void OnAppearing()
{
    base.OnAppearing();
    // Clear selected item on rooms tab to prevent lockup
    ModuleListView.SelectedItem = null;
}

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 67 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
base.CurrentPage = roomsTab;
```

This section of code tries to get application settings. If this fails, there must not be a database present, so a database and table are created and the user is directed to the settings tab to enter settings required to make the application work.

```
//Get app settings
try
{
    List<AppSettings> Settings = await App.appSettings.GetSettingsAsync();
    currentSettings = Settings.Where(x => x.ID == 0).Last();
    currentSettings.ID = 0;
    BindingContext = currentSettings;
}

catch
{
    await App.appSettings.CreateTableAsync();
    List<AppSettings> Settings = await App.appSettings.GetSettingsAsync();
    BindingContext = currentSettings;
}
```

This section of code tries to populate the list of modules. If this fails, there must not be a modules database and a database and table are created.

```
//Get list of rooms
try
{
    modules = await App.modDatabase.GetModulesAsync();
    List<Module> holding = modules.Where(x => x.Location ==
currentSettings.DefaultLocation).ToList<Module>();
    ModuleListView.ItemsSource = holding.Select(x =>
x.Room).Distinct().ToList<string>());
    ModuleListView.BindingContext = modules;
}

catch
{
    await App.modDatabase.CreateTableAsync();
    modules = await App.modDatabase.GetModulesAsync();
    List<Module> holding = modules.Where(x => x.Location ==
currentSettings.DefaultLocation).ToList<Module>();
    ModuleListView.ItemsSource = holding.Select(x =>
x.Room).Distinct().ToList<string>());
    ModuleListView.BindingContext = modules;
}
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 68 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
}
```

This section is erroneous because profiles has not been implemented yet.

```
//Get profiles
try
{
    List<Profile> profiles = await App.profiles.GetProfilesAsync();
    ProfileListView.ItemsSource = profiles;
}
catch
{
    await App.profileDatabase.CreateTableAsync();
    List<Profile> profiles = await App.profiles.GetProfilesAsync();
    ProfileListView.ItemsSource = profiles;
}
```

This block of code gets a list of distinct locations from the list of modules and binds that list to the locations list view. This is the list the user interacts with. This block also sets the item source for the default location picker in the settings tab to the same list of distinct locations.

```
//Get list of locations
LocationListView.BindingContext = modules;
LocationListView.ItemsSource = modules.Select(x =>
x.Location).Distinct().ToList<string>());
defaultLocationPicker.ItemsSource = modules.Select(x =>
x.Location).Distinct().ToList<string>());

try
{
```

This section checks if a broker is configured. If a broker is not configured or not completely configured, the user is directed notified that the broker needs to be configured and taken to the application settings tab. If the broker is configured and already connected. The client is disposed of and instantiated. This is a work around for doze mode. Complying with doze mode would require the application to be completely rearchitected.

```
//Set up Client to work with configured broker
if (currentSettings.BrokerAddress != null && currentSettings.BrokerPassword != null
&& currentSettings.BrokerPort != null && currentSettings.BrokerUserID != null)
{
    if (!App.clientConfigured)
    {
        App.client.SetConnectionInfo(currentSettings);
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 69 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

        App.client.Create();
        App.client.ConnectAsync();
        App.clientConfigured = true;
    }

    if (!App.client.IsConnected() && App.clientConfigured)
    {
        App.client.Dispose();
        App.client = null;
        App.client = new MQtil();
        App.client.SetConnectionInfo(currentSettings);
        App.client.Create();
        App.client.ConnectAsync();
    }

    }
    else
    {
        await DisplayAlert("Alert", "Please Configure a Broker!", "OK");
        base.CurrentPage = settingsTab;
    }
    }
    catch
    {
        await DisplayAlert("Degub", "Ran into an issue setting up MQTT client in
EdgeNetMainPage.cs", "OK");
    }
    }

//Event handler for the edit settings tool-bar item. This enables all of the inputs.
void OnEditSettings_Activated(object sender, ClickedEventArgs e)
{
    brokerAddressEntry.IsEnabled = true;
    brokerPort.IsEnabled = true;
    brokerUserNameEntry.IsEnabled = true;
    brokerPasswordEntry.IsEnabled = true;
    brokerPasswordEntryTwo.IsEnabled = true;
    defaultLocationPicker.IsEnabled = true;
    saveSettingsButton.IsEnabled = true;
}

```

Issued By:	Approved By:	Effective Date:	Page 70 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
//Event handler for saving the app settings.
async void SaveBtn_Clicked(object sender, EventArgs e)
{

    //ToDo - Remove any trailing spaces
    //make sure passwords match
    if (brokerPasswordEntry.Text == brokerPasswordEntryTwo.Text)
    {
        //Try to get data from binding context and save to database
        try
        {
            currentSettings = (AppSettings)BindingContext;
            currentSettings.ID = 0;
            await App.appSettings.SaveSettingsAsync(currentSettings);
        }
        //If the above failed, it's because the table did not exist. Create table and try again.
        catch
        {
            await App.appSettings.CreateTableAsync();
            currentSettings = (AppSettings)BindingContext;
            currentSettings.ID = 0;
            await App.appSettings.SaveSettingsAsync(currentSettings);
        }

        //Disable all app settings inputs
        brokerAddressEntry.IsEnabled = false;
        brokerPort.IsEnabled = false;
        brokerUserNameEntry.IsEnabled = false;
        brokerPasswordEntry.IsEnabled = false;
        brokerPasswordEntryTwo.IsEnabled = false;
        saveSettingsButton.IsEnabled = false;
    }
    else
    {
        //tell the user the passwords do not match
        brokerPasswordEntry.Text = "";
        brokerPasswordEntryTwo.Text = "";
        await DisplayAlert("Passwords Do Not Match", "Please make sure your password and
password confirmation match...", "OK!");
    }
}
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 71 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

}

//Event handler for ListView on RoomList tab.
async void ItemSelected(object sender, SelectedItemChangedEventArgs e)
{
    //make sure an item is selected
    if (e.SelectedItem != null)
    {
        //push module list page and pass data with binding context
        await Navigation.PushAsync(new ModuleListPage(e.SelectedItem.ToString()) {
BindingContext = new Module() });
    }
}

//Event handler for locationList tab
void LocationItemSelected(object sender, SelectedItemChangedEventArgs e)
{
    //Make sure an item is selected
    if (e.SelectedItem != null)
    {
        //Update module list to reflect new location
        //ModuleListView.ItemsSource = modules.Select(x => x.Room).Distinct().ToString()
        base.CurrentPage = roomsTab;
        List<Module> holding = modules.Where(x => x.Location ==
e.SelectedItem.ToString()).ToList<Module>();
        ModuleListView.ItemsSource = holding.Select(x =>
x.Room).Distinct().ToList<string>();
        LocationListView.SelectedItem = null;
    }
}

//event handler for add module toolbar button
async void OnAddModule_Activated(object sender, System.EventArgs e)
{
    //push add module page and set binding context
    await Navigation.PushAsync(new AddModule(true, null) { BindingContext = new
Module() });
}

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 72 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
//This may change. Need to decide if a new view will be used to select from a list of
moduels and configure the profile, or if a
//snapshot of existing modules are a better approach.
async void OnAddRemoveProfile(object sender, EventArgs e)
{
    await Navigation.PushAsync(new AddProfileView() { BindingContext = new Profile()});
}
```

5.2.1.5 LightSwitchControlPage.xaml

The light switch control page is presented to the user when they want to manipulate a dimmable light. In this case that is a PWM peripheral connected to a LED on a dev board. The user is provided with a toggle switch and a slider bar to give binary and gradient control over the light respectively. An image in the center of the view presents the user with a representation of how dim the light is and if it is on or off.

```
<?xml version="1.0" encoding="utf-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="EdgeNet.Views.Interfaces.LightSwitchControlPage"
    Padding="20">
    <ContentPage.ToolbarItems>
```

A button is added to the toolbar to allow the user to edit the module if necessary. This button takes the user to the add module view.

```
        <ToolbarItem x:Name="moduleSettings" Order="Primary" Text="Module Settings"
Activated="ModuleSettingsTBIItem_Activated"/>
    </ContentPage.ToolbarItems>
```

The switch and and associated label needed to be set up on a grid system because a stack layout cannot support multiple columns. The grid is setup inside the stack layout and each item is tied to a grid coordinate.

```
        <StackLayout>
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="auto"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="auto"/>
                </Grid.ColumnDefinitions>
                <Switch x:Name="toggleSwitch" Grid.Column="2" Grid.Row="1"
HorizontalOptions="FillAndExpand" Toggled="OnToggle" IsToggled="{Binding State}"/>
```


Issued By:	Approved By:	Effective Date:	Page 73 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
<Label x:Name="toggleLabel" FontSize="Medium" Grid.Column="1" Grid.Row="1"
Text="" TextColor="#00A79D" HorizontalOptions="FillAndExpand"/>
</Grid>
```

A relative layout was used to place the center graphics. The visual feedback the user sees is accomplished by adjusting the opacity value of the glow image when the dimmer value is changed. An attempt to map these values directly together was made, but to comply with all requirements of the system some processing had to be done before the value was applied.

```
<RelativeLayout>
<Image x:Name="bulb" VerticalOptions="Center" Source="Bulb.png"
HorizontalOptions="Center"/>
<Image x:Name="glow" VerticalOptions="Center" Source="Glow.png"
HorizontalOptions="Center"/>
</RelativeLayout>
```

The dimmer value of the slider is bound to the dimmer value property of the Module object passed to the view when it was created. Notice the range of 0 – 999 matches the range possible in firmware.

```
<Slider x:Name="dimmerSlider" ValueChanged="OnSlider_ValueChanged"
Value="{Binding DimmerValue}" Minimum="0.0" Maximum="999" />
<Image Source="Logo.png" HorizontalOptions="FillAndExpand"/>
</StackLayout>
</ContentPage>
```

5.2.1.6 LightSwitchControlPage.xaml.cs

Code behind for the light switch control page handles some processing of data and manages database activities when something changes, or the page is exited.

```
//Instantiate object to hold current module details
```

```
Module module;
```

```
public LightSwitchControlPage (Module thisModule)
```

```
{
//Assign module details locally accessible variable
this.module = thisModule;
```

```
//Assign module name to view title
Title = module.Name;
```

```
InitializeComponent ();
```

Issued By:	Approved By:	Effective Date:	Page 74 of 91
Travis Cuore	Travis Cuore	10/31/2018	
		Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```
}
```

```
//Things to do when this page appears on screen
```

```
protected override void OnAppearing()
```

```
{
```

```
    base.OnAppearing();
```

```
}
```

```
//Things to do when this page dissappears
```

```
protected override void OnDisappearing()
```

```
{
```

```
    base.OnDisappearing();
```

When the view disappears, the current module is updated with the latest user input before being saved back to the database. Because the tabbed view the user goes back to reloads the database it will get updated there to.

```
    module = (Module)BindingContext;
    App.modDatabase.SaveModuleAsync(module);
}
```

When the slider value is changed, opacity of the glow graphic is updated. A check is done to see if the app is still connected to the broker. If it is, the value is inverted and published.

```
//Things to do when the slider value is changed
```

```
void OnSlider_ValueChanged(object sender, ValueChangedEventArgs e)
```

```
{
```

```
    try
```

```
    {
```

```
        //update opacity of "glow" around bulb graphic
```

```
        glow.Opacity = e.NewValue / 1000;
```

```
        if(App.client.IsConnected())
```

```
        {
```

```
            //publish update to edge module
```

```
            App.client.Pulish(module.MID, 'I', (Math.Abs((999 -
(int)(e.NewValue))))).ToString());
```

```
        }
```

```
    }
```

```
    catch
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 75 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

{
    DisplayAlert("Debug", "Issue while updating slider value", "OK");
}
}

```

This is an event handler for the edit module button placed in the toolbar. It takes the user to the add module page and allows its data to be edited.

```

//take the user to the module settings page
async void ModuleSettingsTBIItem_Activated(object sender, EventArgs e)
{
    await Navigation.PushAsync(new AddModule(false, module) {BindingContext = module
});
}

```

This event handler checks the value of the toggle switch and take appropriate action resulting in data published to the broker.

```

//things to do when the user toggles the object state on screen.
void OnToggle(object sender, ToggledEventArgs e)
{
    try
    {
        //Update values and publish
        if (e.Value)
        {
            module.State = true;
            toggleLabel.Text = "On";
            dimmerSlider.IsEnabled = true;
            glow.IsVisible = true;

            if (App.client.IsConnected())
            {
                App.client.Pulish(module.MID, '2', "1");
            }
        }
        else
        {
            module.State = false;
            toggleLabel.Text = "Off";
            dimmerSlider.IsEnabled = false;
            glow.IsVisible = false;

            if (App.client.IsConnected())

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 76 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

    {
        App.client.Pulish(module.MID, '2', "0");
    }
}
}
catch
{
    DisplayAlert("Debug", "Issue while updating Toggle value", "OK");
}
}

```

5.2.1.7 ModuleListPage.xaml

The module list page view is presented to the user when a room is selected. It is populated with a list of modules where the room and location properties match the location and room already selected.

```

<?xml version="1.0" encoding="utf-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="EdgeNet.Views.ModuleListPage"
    Padding="15">
    <ContentPage.ToolbarItems>

```

The add module button is added to the toolbar because it is appropriate for the given context.

```

        <ToolbarItem x:Name="addModuleTBItem"
            Order="Primary"
            Text="Add Module"
            Activated="AddModuleTBItem_Activated"/>
    </ContentPage.ToolbarItems>
    <StackLayout VerticalOptions="Start" HorizontalOptions="FillAndExpand">
        <ListView x:Name="ModuleListView"
            HorizontalOptions="FillAndExpand"
            VerticalOptions="FillAndExpand"
            ItemSelected="ModuleSelected"
            RowHeight="80">

```

The item template dictates how item data is displayed in each part of the list. In this case, the module name is bound to the text property of the Module object passed to it on creation. The smaller text under the text property is the detail property and it is bound to the type property of the module object passed to the view on creation.

```

        <ListView.ItemTemplate>

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 77 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

        <DataTemplate>
            <TextCell Text="{Binding Name}" Detail="{Binding Type}"
TextColor="#00A79D" DetailColor="DarkSlateGray"/>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
<BoxView VerticalOptions="FillAndExpand"/>

```

The save profile button exists on all list views, but does not function because it is not flushed out yet.

```

        <Button x:Name="saveRoomProfileButton" Text="Save Room Profile"/>
        <Image Source="Logo.png" HorizontalOptions="FillAndExpand"
VerticalOptions="End"/>
    </StackLayout>
</ContentPage>

```

5.2.1.8 ModuleListPage.xaml.cs

//variable to hold selected room

```

public string Room;
public ModuleListPage (string room)
{
    InitializeComponent ();

    //set title to current room
    this.Room = room;
    Title = Room;
}

```

```

protected async override void OnAppearing()
{
    base.OnAppearing();
}

```

When this view appears to the user, a list of modules for the selected room is loaded. If this list cannot be loaded a database must not exist so the system attempts to make one. However, it should be noted that this page cannot be reached without having something in the database.

```

    //try to load list of modules for current room
    try
    {
        List<Module> modules = await App.modDatabase.GetModulesAsync();
        ModuleListView.ItemsSource = modules.Where(x => x.Room ==
Room).ToList<Module>();
    }
}

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 78 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

    }
    //if modules cannot be loaded, create table and set
    catch
    {
        await App.modDatabase.CreateTableAsync();
        List<Module> modules = await App.modDatabase.GetModulesAsync();
        ModuleListView.ItemsSource = modules.Where(x => x.Room ==
Room).ToList<Module>());
    }
}

```

The event handler for a selected module will first check to see if an item is selected because the event handler looks at a change in selection state. If an item is selected, the type of module is checked and connectivity to the broker is checked before passing selected module data to the light switch control page view. Binding context is set to the selected module at this time.

```

//Event handler for tap on list
async void ModuleSelected(object sender, SelectedItemChangedEventArgs e)
{
    //make sure an item is selected
    if (e.SelectedItem != null)
    {
        Module currentModule = (Module)e.SelectedItem;
        if (currentModule.Type == "Dimming Light")
        {
            while (!App.client.IsConnected())()
            {
            }

            //push the module interface page and pass data for the selected item
            await Navigation.PushAsync(new
LightSwitchControlPage((Module)e.SelectedItem) { BindingContext =
(Module)e.SelectedItem });
        }
    }
}

```

The add module button is added to the toolbar because it is appropriate for the given context.

```

//event handler for the add module toolbar button
async void AddModuleTBIItem_Activated(object sender, System.EventArgs e)
{
    //push the add module page and set binding context to Module

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 79 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

        await Navigation.PushAsync(new AddModule(true, null) {BindingContext = new
Module()});
    }

```

5.2.2 Database Wrappers

These wrappers are all abstractions of the sqlite library. This approach to implementing a library makes it easy to automate tasks in a way that is context sensitive to the application being written. In this case, two databases were needed. One for application settings, and one for modules. Technically the application could do without the modules database, but it improves responsiveness and speed since the application doesn't need to go through discovery, or poll devices on start. It just loads the last known state for each module and polls only when a module is loaded.

All of these wrappers require the following using directives to function properly:

```

using SQLite;
using System.Collections.Generic;
using System.Threading.Tasks;

```

Because all these classes are identical in their operation, only one will be commented for this document. All classes will be presented for the sake of completeness.

5.2.2.1 AppSettingsDatabase.cs

```

public class AppSettingsDatabase
{
    readonly SQLiteAsyncConnection database;

```

When the database is instantiated, an attempt is made to get a path for it with the local interface. As the constructor argument, the local interface is invoked and the local database path is returned.

```

    public AppSettingsDatabase(string dbPath)
    {
        database = new SQLiteAsyncConnection(dbPath);
    }

```

All of these methods are public tasks. This is done to call the sqlite library methods as async. This method creates a table in the database which basically initializes it to a specific structure. All other tasks are done in the same way and for the same reasons. Their names are indicative of what they do, so commenting was not necessary for this file.

```

    public Task<CreateTablesResult> CreateTableAsync()
    {

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 80 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

    return database.CreateTableAsync<AppSettings>();
}
public Task<List<AppSettings>> GetSettingsAsync()
{
    return database.Table<AppSettings>().ToListAsync();
}

public Task<AppSettings> GetSettingAsync(int id)
{
    return database.Table<AppSettings>().Where(i => i.ID == id).FirstOrDefaultAsync();
}

public Task<int> SaveSettingsAsync(AppSettings appSettings)
{
    if (appSettings.ID != 0)
    {
        return database.UpdateAsync(appSettings);
    }
    else
    {
        return database.InsertOrReplaceAsync(appSettings);
    }
}

public Task<int> DeleteSettingsAsync(AppSettings appSettings)
{
    return database.DeleteAsync(appSettings);
}
}

```

5.2.2.2 ModuleDatabase.cs

```

public class ModuleDatabase
{
    readonly SQLiteAsyncConnection database;

    public ModuleDatabase(string dbPath)
    {
        database = new SQLiteAsyncConnection(dbPath);
    }

    public Task<CreateTablesResult> CreateTableAsync()
    {

```


Issued By:	Approved By:	Effective Date: 10/31/2018	Page 81 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

    return database.CreateTableAsync<Module>();
}

public Task<List<Module>> GetModulesAsync()
{
    return database.Table<Module>().ToListAsync();
}

public Task<Module> GetModuleAsync(int id)
{
    return database.Table<Module>().Where(i => i.ID == id).FirstOrDefaultAsync();
}

public Task<int> SaveModuleAsync(Module module)
{
    if(module.ID != 0)
    {
        return database.UpdateAsync(module);
    }
    else
    {
        return database.InsertAsync(module);
    }
}

public Task<int> DeleteModuleAsync(Module module)
{
    return database.DeleteAsync(module);
}

```

5.2.3 MQTT Wrapper

The MQTT wrapper is a class wrapped around the methods provided by the OpenNetCF.MQTT client library. This wrapper accomplishes two things. First, it allows for a client to be instantiated as an object with additional properties. Second, it allows for applications settings to be automatically passed into the client object on creation.

The following using directives are required for the MQTT wrapper:

```

using System;
using System.Text;
using OpenNETCF.MQTT;
using System.Threading;
using System.Diagnostics;

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 82 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

using System.Threading.Tasks;

5.2.3.1 MQtil.cs

public class MQtil

```
{
    public AppSettings connectionInfo;
    private MQTTClient client;

    public MQtil()
    {
    }

    public void UpdateConnectionInfo(AppSettings newConnectionInfo)
    {
        connectionInfo = newConnectionInfo;
    }

    public void Create()
    {
        try
        {
            client = new MQTTClient(connectionInfo.BrokerAddress,
int.Parse(connectionInfo.BrokerPort));
        }
        catch
        {
            Debug.WriteLine("Wrapper could not creat new client");
        }
    }

    public void SetConnectionInfo(AppSettings newInfo)
    {
        try
        {
            connectionInfo = newInfo;
        }
        catch
        {
        }
    }
}
```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 83 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

        Debug.WriteLine("Wrapper could not set new connection info");
    }
}

public void SetReconnectPeriod()
{
    client.ReconnectPeriod = 1000;
}

public async void ConnectAsync()
{
    try
    {
        if (!client.IsConnected)
        {
            await client.ConnectAsync("Android Test", connectionInfo.BrokerUserID,
connectionInfo.BrokerPassword);
            //await WaitForConnection();
        }
    }
    catch
    {
        Debug.WriteLine("Wrapper could not connect to broker");
    }
}

public void Dispose()
{
    try
    {
        client.Dispose();
    }
    catch
    {
        Debug.WriteLine("Wrapper could not Dispose of client");
    }
}

public void Disconnect()
{

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 84 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

try
{
    client.Disconnect();
}
catch
{
    Debug.WriteLine("Wrapper could not Disconnect");
}
}

public void RegisterMessageRecievedEventHandler()
{
    try
    {
        // hook up the MessageReceived event with a handler
        client.MessageReceived += (topic, qos, payload) =>
        {
            //ToDo write switch case to handle different incomming events
            Debug.WriteLine("MQTT: " + Encoding.UTF8.GetString(payload));
        };
    }
    catch
    {
        Debug.WriteLine("Wrapper could not register event handler");
    }
}

public string GetConnectionState()
{
    return client.ConnectionState.ToString();
}

public void RegisterDisconnectEventHandler()
{
    try
    {
        // hook up the MessageReceived event with a handler
        client.Disconnected += (object sender, EventArgs e) =>
        {
            //ToDo write switch case to handle different incomming events

```

Issued By:	Approved By:	Effective Date:	Page 85 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

        client.Connect("Android Test", connectionInfo.BrokerUserID,
connectionInfo.BrokerPassword);
        WaitForConnection();
    };
}
catch
{
    Debug.WriteLine("Wrapper could not register event handler");
}
}

public void Subscribe(string topic, QoS qs)
{
    try
    {
        client.Subscriptions.Add(topic, qs);
    }
    catch
    {
        Debug.WriteLine("Wrapper could not subscribe to topic");
    }
}

public void Unsubscribe(string topic)
{
    try
    {
        client.Subscriptions.Remove(topic);
    }
    catch
    {
        Debug.WriteLine("Wrapper could not unsubscribe from topic");
    }
}

public void Publish(string MID, char CMD, string payload)
{

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 86 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

try
{
    client.Publish("test" + "/" + MID, CMD + payload, QoS.FireAndForget, false);
}
catch
{
    Debug.WriteLine("Wrapper could not publish to topic");
}
}

public bool IsConnected()
{
    if(client.IsConnected)
    {
        return true;
    }
    else
    {
        return false;
    }
}

private Task WaitForConnection()
{
    try
    {
        int i = 0;
        while (!client.IsConnected)
        {
            Thread.Sleep(250);

            if (i++ > 40)
            {
                return Task.CompletedTask;
            }
        }
    }
    catch
    {
        Debug.WriteLine("Wrapper got tired of waiting for connection (could not connect)");
    }
}

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 87 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

        return Task.CompletedTask;
    }

    return Task.CompletedTask;
}

```

5.2.4 Data Objects

Both of the following objects follow the same structure. However, it should be noted that only the module object auto-increments the primary key. This is because we only ever want one version of AppSettings in the table.

5.2.4.1 AppSettings.cs

```
using SQLite;
```

```
namespace EdgeNet
{
    public class AppSettings
    {
        [PrimaryKey]
        public int ID { get; set; }
        public string BrokerAddress { get; set; }
        public string BrokerPort { get; set; }
        public string BrokerUserID { get; set; }
        public string BrokerPassword { get; set; }
        public string DefaultLocation { get; set; }
    }
}

```

5.2.4.2 Module.cs

```
namespace EdgeNet
{
    public class Module
    {
        [PrimaryKey, AutoIncrement]
        public int ID { get; set; }
        public string MID { get; set; }
        public string Type { get; set; }
        public string Name { get; set; }
        public string Room { get; set; }
        public string Location { get; set; }
    }
}

```

Issued By:	Approved By:	Effective Date:	Page 88 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

public bool State { get; set; }
public double DimmerValue { get; set; }
}

```

5.2.5 Local Resource Interface

The following two files are required to implement local interfaces needed for shared code to access platform specific resources such as storage.

5.2.5.1 ILocalFileHelper.cs

This file implements the shared code hook for localized code to extend.

```

using System;
using System.Collections.Generic;
using System.Text;

namespace EdgeNet
{
    public interface ILocalFileHelper
    {
        string GetLocalFilePath(string fileName);
    }
}

```

5.2.5.2 LocalFileHelper.cs

This file implements a localized interface by extending the shared code class of ILocalFileHelper. This class returns the filepath of the requested database.

```

using System;
using System.IO;
using Xamarin.Forms;
using EdgeNet.iOS;

[assembly: Dependency(typeof(LocalFileHelper))]

namespace EdgeNet.iOS
{
    public class LocalFileHelper : ILocalFileHelper
    {
        public string GetLocalFilePath(string fileName)
        {
            string docFolder = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            string libFolder = Path.Combine(docFolder, "..", "Library", "Databases");
            if (!Directory.Exists(libFolder))

```


Issued By:	Approved By:	Effective Date:	Page 89 of 91
Travis Cuore	Travis Cuore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

    {
        Directory.CreateDirectory(libFolder);
    }
    return Path.Combine(libFolder, fileName);
}
}
}

```

5.2.6 App.cs

This file sets up all objects required for the application to function. It reads databases and sets up an interface for accessing them.

`namespace EdgeNet`

```

{
    public partial class App : Application
    {

        //instantiate database objects
        public static ModuleDatabase moduleDatabase;
        public static AppSettingsDatabase appSettingsDatabase;
        public static AppSettings currentSettings;
        public static ProfileDatabase profileDatabase;
        public List<AppSettings> Settings;
        public static MQtil client = new MQtil();
        public static bool clientConfigured = false;

        public App()
        {
            InitializeComponent();

            //Make database connections
            moduleDatabase = new
ModuleDatabase(DependencyService.Get<ILocalFileHelper>().GetLocalFilePath("Modules.db3
"));
            appSettingsDatabase = new
AppSettingsDatabase(DependencyService.Get<ILocalFileHelper>().GetLocalFilePath("AppSetti
ngs.db3"));

            MainPage = new NavigationPage(new EdgeNetMainPage() { BindingContext = new
Module().Room });
        }

        //An object to get the module database from other forms
    }
}

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 90 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

public static ModuleDatabase modDatabase
{
    get
    {
        if (moduleDatabase == null)
        {
            moduleDatabase = new
ModuleDatabase(DependencyService.Get<ILocalFileHelper>().GetLocalFilePath("Modules.db3
"));
        }
        return moduleDatabase;
    }
}

//An object to get the app settings database from other forms
public static AppSettingsDatabase appSettings
{
    get
    {
        if (appSettingsDatabase == null)
        {
            appSettingsDatabase = new
AppSettingsDatabase(DependencyService.Get<ILocalFileHelper>().GetLocalFilePath("AppSetti
ngs.db3"));
        }
        return appSettingsDatabase;
    }
}

public static ProfileDatabase profiles
{
    get
    {
        if(profileDatabase == null)
        {
            profileDatabase = new
ProfileDatabase(DependencyService.Get<ILocalFileHelper>().GetLocalFilePath("Profiles.db3")
);
        }
        return profileDatabase;
    }
}

```

Issued By:	Approved By:	Effective Date: 10/31/2018	Page 91 of 91
Travis Cucore	Travis Cucore	Document No.: 0001	Version: 01

Internet of Things Proof of Concept

```

    }
}

protected override void OnStart ()
{
}

protected override void OnSleep ()
{
}

protected override void OnResume()
{
}
}
}

```

5. CONCLUSIONS AND RECOMMENDATIONS

The EdgeNet IoT platform provides a complete framework for implementing an end-to-end IoT solution. Its architecture hardens the platform against loss of connectivity and make maintenance, implementation and customization easier.

The embedded client can be implemented as a simple library with minimal configuration needing only to be tied into the WiFi stack for triggering. The mobile application is easy to use, and can be rebranded and/or tailored for a wide range of practical uses.

References

- [1] Joeyzh, Artist, *Use MQTT to Upload and Listen to Ameba*. [Art]. Instructables, 2018.
- [2] Developer.Android.com, "Distribution Dashboard | Android Developers," 31 October 2018. [Online]. Available: <https://developer.android.com/about/dashboards/>. [Accessed 31 October 2018].
- [3] OASIS, *MQTT Version 3.1.1 Plus Errata 01*, A. B. a. R. Gupta, Ed., 2015, 2015.
- [4] Microchip, *Wi-Fi Network Controller Software Design Guide*, Microchip, 2018.
- [5] Microchip, *IEEE 802.11 b/g/n SmartConnect IoT Module*, Microchip, 2018.
- [6] Microchip, *SAM D21E / SAM D21G / SAM D21J Datasheet Complete*, Microchip, 2018.